WO2004100556

Publication Title:

DATA PROCESSING

Abstract:

Abstract of WO2004100556

Systems and methods for coding data, particularly image data such as video data, are described. In particular, a method of coding data having at least first, second and third portions is described, wherein the method includes forming an estimate of the accuracy of a prediction of the second portion based on the first portion and selectively predicting the third portion based at least on the second portion and the estimate of accuracy. Other related aspects of coding data by prediction, for example frames or fields of video data, are described, including methods of decoding data and methods of error correction and motion vector tracing.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 November 2004 (18.11.2004)

PCT

(10) International Publication Number
WO 2004/100556 A2

(51) International Patent Classification^7: H04N 7/32, G06T 9/00

(21) International Application Number: PCT/GB2004/001987

(22) International Filing Date: 7 May 2004 (07.05.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0310495.7 7 May 2003 (07.05.2003) GB

(71) Applicant *(for all designated States except US)*: **BRITISH BROADCASTING CORPORATION** [GB/GB]; Broadcasting House, London WC1A 1AA (GB).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **DAVIES, Thomas** [GB/GB]; c/o British Broadcasting Corporation, Research Department, Kingswood Warren, Tadworth, Surrey KT20 6NP (GB). **GRAHAM, Thomas** [GB/GB]; 'Anderida', Ridge Close, Nutley, Uckfield, East Sussex TN2 3HA (GB). **TUDOR, Philip** [GB/GB]; St. Marthas Cottage, Barnett Lane, Wonersh, Guildford, Surrey GU5 0SA (GB). **BORER, Timothy** [GB/GB]; 3 Ontario Close, Smallfield, Surrey RH6 9RA (GB).

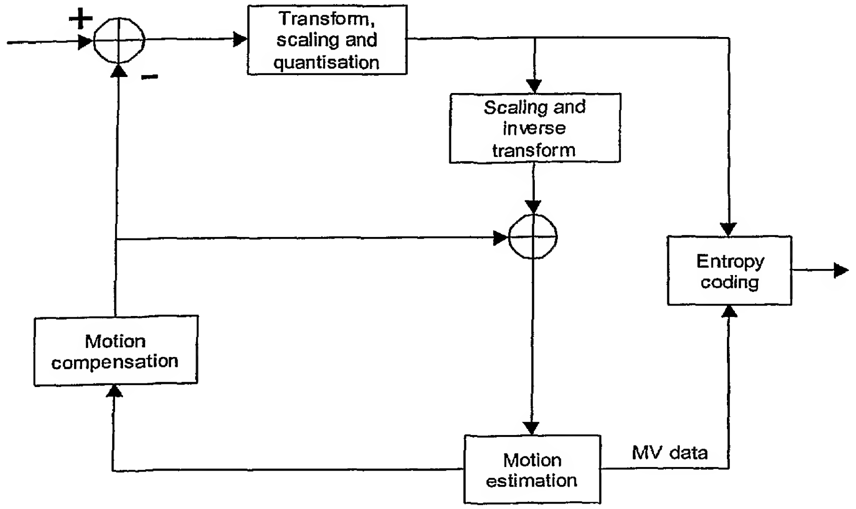
(74) Agents: **KAZI, Ilya** et al.; Mathys & Squire, 100 Gray's Inn Road, London WC1X 8AL (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI,

[Continued on next page]

(54) Title: DATA PROCESSING


+
-
Transform, scaling and quantisation
Scaling and inverse transform
Entropy coding
Motion compensation
Motion estimation
MV data

(57) **Abstract:** Systems and methods for coding data, particularly image data such as video data, are described. In particular, a method of coding data having at least first, second and third portions is described, wherein the method includes forming an estimate of the accuracy of a prediction of the second portion based on the first portion and selectively predicting the third portion based at least on the second portion and the estimate of accuracy. Other related aspects of coding data by prediction, for example frames or fields of video data, are described, including methods of decoding data and methods of error correction and motion vector tracing.

WO 2004/100556 A2

SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Data Processing

The present invention relates to coding of data, particularly media or image data, particularly for motion video. However, aspects of the invention may be applied to other types of data.

A large number of methods for encoding data are known. Some methods use prediction of a portion of the data based on other portions of the data. For example, encoding of motion video using MPEG, a video frame may be predicted based on the content of a preceding frame and estimated motion within the frame. It is then only necessary to code the difference between the prediction and the actual data. In sequences which can be easily predicted, the amount of difference information required for satisfactory reproduction may be significantly less than the original data, leading to beneficial compression. However, a drawback is that, particularly for sequences which are difficult to predict, a significant amount of information may be needed to convey the information used in prediction (e.g. motion vectors) and the resulting predictions may still not be very accurate so the saving in data between the original data and the difference information is relatively small. There may even be cases where the prediction gives a worse starting point for determining difference information than no prediction at all.

In summary, predictive methods have two main drawbacks. Firstly, the information required for a successful prediction may become comparable in volume to the data thereby negating the benefit of prediction in data compression. Secondly, the prediction may fail significantly leading to little or no saving in data. Our earlier UK Patent Application No. 0122526.7 seeks to alleviate the first problem by using prediction based on data that will be available to a recipient of the coded data so that the prediction can be recreated by the recipient without having to communicate all of the parameters used in the prediction. However, there is still room for improvement on this technique.

It should be noted that, while predictive techniques have been described above in the context of inter-frame coding, where prediction is based on preceding frames of a video sequence (typically by predicting motion of objects between frames), predictive coding techniques can also be used within a single field or frame of video information (intra-frame prediction) based on information available elsewhere in the frame. Prediction can also be used with data which are not necessarily image data and which need not necessarily be structured, for example text. In many cases where data have inter-dependencies, prediction can be used to reduce the amount of information that needs to be transmitted to recreate the data. By way of further background history, GB-A-2,195,062 discloses a predictive coding method for a television signal. Aspects of the present invention are generally concerned with predictive coding techniques.

It will be noted that video sequences may be distributed in essentially two ways. Firstly, an entire video sequence may be transmitted as an entity, for example as a computer file. In such a case, the entire sequence will be available to an entity attempting to display the sequence, and compression techniques may be developed based on complete knowledge of the sequence and relying on the complete coded sequence being available to a decoder in order to display the video. Such techniques may be highly efficient but a drawback is that display of the sequence is only possible if the entire sequence is available and this may require large or prohibitive amounts of memory. This situation is not generally applicable to real time transmission of video sequences or to large sequences as playback cannot occur until the entire file is received. In a second possibility, the video is compressed in such a way that a sequence can be displayed when only a portion is available, and generally it is only necessary to store a few frames of data at a time in order to decode the sequence; broadcast MPEG II is in this latter category which will herein be termed a "serially accessible" video sequence. Another criteria is that the sequence is substantially 'randomly accessible', that is the sequence that can be accessed at regular intervals, without requiring the sequence to be received from the start (although there may be delays before an accessible key point). The present invention seeks to provide a coding method suitable for use in compressing video to provide a serially accessible sequence and preferably substantially randomly accessible sequence. However, the invention is applicable to other data and the sequence can be accessed in a parallel fashion if desired.

Aspects of the invention are set out in the independent claims and preferred features are set out in the dependent claims.

In a first aspect, the invention provides a data processing method for processing a stream of data to be coded using prediction, the stream comprising at least first, second and third portions, in which the second portion may be coded using prediction based on the first portion and the third portion may be coded using prediction based on at least the second portion, the method comprising:
forming an estimate of accuracy of a prediction of the second portion based on the first portion; and
selectively predicting the third portion based on at least the second portion and the estimate of accuracy of the prediction of the second portion.

Pursuant to the invention it has been appreciated that the effectiveness of an earlier prediction often gives a good indication of the likely effectiveness of a subsequent prediction for many types of real world data. It is very important to note that this method will *not* always give an optimum result. On the contrary, a much better "estimate" of the accuracy of prediction of the third portion can be obtained by actually doing the prediction as the necessary information is available at the coder. In many cases, as might be expected, the estimate of accuracy based on the estimate of accuracy of the previous prediction may turn out to be wrong, for example around transitions in the data. However, surprisingly, pursuant to the invention it has been appreciated that nonetheless the estimate of accuracy based on previous prediction is found to be helpful overall. A particular advantage is that the estimate can be made with data that will be available to a decoder which is receiving the signal serially and has only received the first and second portions.

The third portion will typically comprise an individual coefficient but may comprise a set of coefficients. The method will typically be repeated a plurality of times for subsequent coefficients, wherein each coefficient is selectively predicted based on the success (actual or estimated) of a previous prediction. The method may be applied iteratively, selectively predicting subsequent portions based on the success of preceding predictions. In repeating the method iteratively, the second and third portions of a first selective prediction may become respectively the first and second portions of a second selective prediction, to predict a new third portion.

Preferably the estimate is based only on data that will be available to a decoder which is receiving the signal serially.

The method may be applied to a third portion which comprises an individual coefficient (preferably) or to a set of coefficients.

Selectively predicting the third portion may comprise comparing the magnitude of the second portion (or elements thereof) to the magnitude of the difference between the second and first portions (or the difference between one or more elements of the second and first portions).

The third portion may be predicted only if the second portion is substantially greater than said difference between the second and first portions. The second portion may be required to be at least a given multiple of the said difference between the second and first portions. It has been found advantageous if said multiple is at least two, preferably at least about four, more preferably about 6.

In place of a linear multiplication, a non-linear scaling function may be applied which gives multipliers differing magnitudes by differing amounts, for example a higher "standard of proof" (ratio of difference to coefficient value) may be required for prediction of small coefficients on the basis that noise may be more significant or conversely a higher standard may be applied to larger coefficients on the basis that prediction using a large (wrong) absolute value may result in greater coding error than predicting based on a small value. A polynomial function may be used, with coefficients empirically determined for a given set of data – the coefficients may be pre-set or configurable or adaptively determined. In practice, however, it is found that a simple linear scaling function is highly effective and simple to configure and implement.

In a simple but highly effective implementation, if the third portion is predicted, the second portion is used as a prediction.

Alternatively, prediction may be based on several preceding samples. Thus, where the third portion comprises a single coefficient, the decision whether or not to predict may be made on the basis of success of prediction of a second portion comprising a single coefficient but the prediction itself may be made on the basis of a plurality of coefficients, based on the (recent) history of the sample. The decision whether or not to predict may also be based on the success of prediction of a number of coefficients rather than a single coefficient and the second portion may differ in size from the third portion. Similarly, the first portion may comprise a number of coefficients from the recent history of the data.

Following selectively predicting, either the third portion or the difference between the third portion and a prediction thereof is preferably coded.

The data may be processed by a hierarchical coding scheme in which data is coded in successive layers, wherein the third portion comprises data from a higher layer and the second portion comprises corresponding data from at least one lower layer.

Selective prediction of the third portion may comprise a prediction based on the data preceding the third portion if a parameter multiplied by a measure of the magnitude of the difference between the second portion and a prediction of the second portion based on the data preceding the second portion is less than a measure of the magnitude of the second portion, and a default prediction otherwise. Preferably, where the third portion comprises one or more numerical coefficients, the default prediction is zero. For non numerical coding, a default symbol may be used. Preferably the default is based on a set value for the expected mean of the samples (or may be based on the actual mean of the last predetermined number of samples although if this number of samples is too high, this may introduce problems for decoder synchronization). The parameter may have a value of at least two.

The parameter may be set to a fixed value. Alternatively, the value of the parameter is adjusted based on dynamic analysis of the data. The parameter may be adjusted based on at least one of:-

a) The prior inputs to the selective prediction method;
b) The prior outputs of the selective prediction method;
c) The prior outputs of the prediction based on the data preceding the portion to be coded ;
d) The prior outputs of the comparison.

If the adaptive algorithm adapts over too long a period, this may present problems for decoder synchronization. However, this can be alleviated by restarting adaptation at periodic intervals and/or communicating a default adaptation context at intervals. Typically, where the method is used to code portions comprising pixels of a frame, adaptation may be re-started each frame. However, it does not require much bit rate to signal with each frame a few coefficients which give an adaptation context if desired and this may provide some advantage in some applications.

The method of prediction may be dynamically adjusted based on the results of prediction for preceding data. The effectiveness of the selective prediction method is compared with the effectiveness of the basic prediction method.

In a further aspect but closely related aspect, the invention provides a method of selectively predicting a current portion of data based on preceding data comprising selectively predicting a current portion of data x(i) in accordance with the formula:

$$P_1(S(i-1)) = \begin{cases} P(S(i-1)) \text{ if } \lambda \cdot C(P(S(i-2)) - x(i-1)) < C(x(i-1)) \\ 0 \text{ otherwise} \end{cases}$$

wherein

P_1(S(i-1)) represents the selective prediction of the current portion of data based on at least a part of the set of data preceding the current portion;

P(S(i-1)) represents a prediction of the current portion of data based on at least a part of the set of data preceding the current portion;

P(S(i-2)) represents a prediction of the preceding portion of data based on at least a part of the set of data preceding the preceding portion;

C represents a measure of cost of transmitting or storing the data;

x(i-1) represents the preceding portion of data;

λ represents a parameter, normally at least two.

In place of a multiplication parameter, a scaling function may be applied, as mentioned above.

Each portion of data may comprise a single coefficient. Each prediction may simply comprise the preceding coefficient. The cost measure may simply comprise the magnitude of the coefficient. The cost measure preferably comprises a measure of the entropy of the coefficient (which relates to the "cost" (in terms of bits required) to encode it).

One or both of the cost measure, the method of prediction and the parameter may be adapted in response to successive portions of data.

Portions to be coded are preferably quantised prior to prediction; this facilitates the decoding process if quantisation is used.

Alternatively, portions to be coded may be unquantised and the residual data after selective prediction is quantised. In this case, preferably the coefficients used for prediction are reconstructed coefficients, on which inverse quantisation and prediction have been performed.

In the case of adaptive modification of prediction method or cost measure, adjustment is preferably based on reconstructed coefficients. This facilitates repeating the operation at a decoder.

In a further aspect, the invention provides a method of coding a video picture in which the picture is partitioned into at least two portions, the method comprising predicting elements of the second portion by displacing elements of the first portion and coding the differences between elements of the second portion and the respective predictions thereof.

Wavelet coding is known not to code diagonal edges well, and this method could be used to shuffle lines so that elements of the lines about the edges are aligned vertically by said displacing before filtering.

The two portions may correspond to first and second times, in which case displacing elements may comprise displacing elements based on an estimate of motion between the first and second times.

Preferably elements of the first portion are positioned at spatially different but substantially adjacent positions to elements of the second portion. Preferably the first and second portions are interleaved. The portions most conveniently comprise alternating picture lines, preferably vertically separated, but horizontal and other separation may be used additionally or alternatively.

In a preferred implementation, the first and second portions comprise respectively first and second fields of an interlaced video picture frame.

In a preferred implementation of this aspect, there is provided a method of coding an interlaced video picture frame comprising at least first and second fields, each field comprising a plurality of picture lines, the lines of the first field interleaving with lines of the second field, the first and second fields corresponding to respective first and second times, the method comprising:
predicting elements of the second field based on the first field, wherein predicting comprises shifting elements of the first field along the direction of said picture lines based on an estimated component of motion along said lines between said first and second times, whereby elements of the second field are predicted based on elements of the first field which are estimated to be aligned;
coding the second field following prediction.

This is found to give an effective method of efficiently coding interlaced video, better than simple filtering which takes advantage of the similarities between adjacent fields but without requiring complex motion compensation to reconstruct a progressive picture.

Typically, the lines are horizontal and only horizontal motion is estimated. Estimating horizontal motion only has the benefit that the motion estimation process can be greatly simplified, requiring little computational power. In particular, only a one-dimensional search is required to find a match and this may be an order of magnitude faster than a traditional two-dimensional search for a motion vector. It may only be necessary to store a few lines of picture to perform the motion estimation, for example if prediction is based simply on the preceding line, only two lines are required for the motion estimation and this can efficiently be implemented in hardware if required.

However, vertical motion may also be estimated. Conventional two dimensional motion estimation may be used. Alternatively, vertical motion estimation may be first performed only to integer line accuracy to index a vertical line to be used as the basis of horizontal motion estimation and this two-stage process may be faster than a conventional two-dimensional estimate.

Elements of a field may be predicted using a single element from the other field as a basis for prediction, and this has numerous advantages in terms of algorithm simplicity and efficiency. However, multiple elements may be used, using long filters if desired (and these may extend in both the vertical and the horizontal direction if desired) and motion may be estimated to sub-pixel accuracy in both vertical and horizontal directions. In practice, improvements in prediction accuracy from the greater complexity may be outweighed by the complexity of the prediction method.

Advantageously, elements of the first field are used both as a basis for estimating a component of apparent motion between the first and second fields and as a basis for predicting values of elements of the second field.

In a closely related aspect, the invention provides a method of coding an interlaced video frame comprising first and second fields, the method comprising predicting elements of the second field based on elements of the first field which are displaced horizontally in accordance with estimated horizontal components of motion between the first and second fields and coding the second field following prediction.

Only a single estimate of horizontal motion may be used for each picture line; this simplifies problems of multiple predictors pointing to the same pixel but gives effective results. Having multiple predictors may cause difficulties when coding the *first* field by the method described herein.

Each element of the second field is preferably associated with a predictor comprising an element of the first field from a corresponding picture line at a position along the corresponding line corresponding to the position along the line of the element of the second field to be predicted plus an offset based on estimated motion.

The corresponding picture line may simply be the line above. Alternatively, it may comprise another line indexed by a first stage of vertical motion estimation, as described above.

Advantageously, the interlaced picture is coded by a hierarchical or recursive coding method after prediction. Most preferably, the picture is coded by a wavelet coding method following prediction.

Preferably a difference between each element of the second field and a corresponding element of the first field is coded. Preferably the corresponding element of the first field comprises an element of the first field shifted based on a component of estimated motion. The difference may be used to provide a high-pass output of a wavelet coding filter, preferably wherein half the difference is coded. A low pass output of the wavelet coding filter may comprise the corresponding a component if the estimated prediction predicts no elements on a line or, if one or more elements on the second line are predicted, the average of the corresponding element of the first field and the predicted element or elements of the second field.

Although this aspect works well with horizontal motion compensation alone, vertical motion compensation could also be incorporated
Each element preferably comprises an individual pixel. However, each element may comprise a group of pixels. The frame may be coded without further motion estimation. Further wavelet decomposition may be performed without further motion estimation.

In a further aspect, the invention provides a method of hierarchical coding of data in which data is coded by a coding scheme into at least parent and child coding levels, wherein parent and child levels are both quantised, the method comprising selecting quantisation parameters for at least a portion of the child coding level based on the quantisation parameters for at least a corresponding portion of the parent coding level.

In this way, coding of child level(s) is simplified and less information needs to be communicated to a decoder to reconstruct the child levels. A minor drawback in certain cases is that the child levels cannot be reconstructed independently from the parent levels, but this will typically not be problematic.

The coefficients in the parent coding level may be partitioned and the partitioning of the child level may be based on the partitioning of the parent level. Preferably, partitioning of the parent level is based on the quantised coefficients of the parent level. Preferably, the parent level is communicated to a decoder prior to the child level.

The principle is not limited to quantisation classes, but could be used for other coding parameters, for example for defining contexts for entropy coding. The classification could be used for defining quantisers as well as quantisation classes. If the quantiser values themselves are determined implicitly, this avoids the need to signal them to the decoder.

A useful development is to classify coefficients in the parent level into subsets with different variances or other statistical properties, for example into zero and non-zero coefficients, or into zero, small, and large coefficients. Local properties can also be taken into account, for example local masking so that a class of coefficients defined in this way can be quantised more heavily if the results are likely to be masked, or less heavily in a region of interest.

Implicit classification into subsets, although implicit, may be assisted by additional signalling from the encoder.

It is important to note that this aspect of the invention, although applied to video in the embodiment, can be applied to other signals and can be applied to coding of multiple levels (with children and grandchildren) and to coding in which each parent has a number of children coefficients (for example in audio coding using wavelet transforms in which each parent coefficient has two child coefficients). Similarly, although wavelet transforms are advantageous, other transforms in which parent-child relationships can be defined may be used, for example Block transforms, Lapped-Orthogonal Transforms and others.

In a further aspect, the invention provides a method of coding a sequence of pictures comprising:
selecting only a subset of the sequence of pictures for communication leaving a subset of residual pictures;
interpolating from the subset of pictures for communication according to an interpolation algorithm to create at least one interpolated picture corresponding substantially to one of the subset of residual pictures;
encoding adjustment information based on a difference between the at least one interpolated picture and the corresponding residual picture;
communicating the subset of pictures for communication together with the adjustment information.

In this way, a higher frame rate can be recreated from a signal which is essentially a lower frame rate using interpolation at a decoder and then adjustment of the interpolated frames (which may require considerably less information to achieve than transmission of the frames by a conventional method).

Preferably the subset of pictures is encoded.

The sequence of pictures has a first frame rate and the subset of pictures preferably comprises a sequence of pictures having a lower frame rate than the first frame rate.

Pictures may be dropped (i.e. be selected as residual pictures) at substantially regular intervals to reduce the frame rate. Additionally or alternatively pictures may be dropped based on picture information content, for example low information content pictures may be dropped, so that little adjustment information is required to recreate them. Pictures may be dropped to control the output frame rate and/or to control the output bit rate. Both selection of pictures and coding parameters may be adjusted to control the output bit rate.

The method typically further comprises communicating the output to be received by a decoder arranged to perform interpolation in accordance with the interpolation algorithm.

A complementary aspect provides a method of reconstructing a sequence of pictures comprising:
receiving a sequence of pictures to produce a sequence of received pictures;

interpolating at least one picture from the received pictures according to a predetermined interpolation algorithm;
receiving adjustment information and applying the adjustment information to the or each interpolated picture to modify the or each interpolated picture;
outputting a sequence of pictures comprising the received pictures and the modified interpolated pictures.

The method may include decoding the sequence of pictures received to provide received pictures.

In an aspect relating to the overall system, there is provided a method of communicating a sequence of pictures comprising:
at a coder,
selecting only a subset of the sequence of pictures for communication leaving a subset of residual pictures;
interpolating from the subset of pictures for communication according to an interpolation algorithm to create at least one interpolated picture corresponding substantially to one of the subset of residual pictures;
encoding adjustment information based on a difference between the at least one interpolated picture and the corresponding residual picture;
communicating the subset of pictures to a decoder together with the adjustment information; and
at the decoder,
receiving the subset of pictures to produce a sequence of received pictures;
interpolating at least one picture from the received pictures according to the interpolation algorithm;
receiving the adjustment information and applying the adjustment information to the or each interpolated picture to modify the or each interpolated picture;
outputting a sequence of pictures comprising the received pictures and the modified interpolated pictures.

In all the related aspects, the interpolation algorithm preferably comprises interpolating based only on information available for one or more pictures other than the picture to be interpolated. Pictures other than the picture to be interpolated may comprise pictures encoded using prediction based on motion vectors, for example according to an MPEG II or similar scheme. Alternatively, pictures may be coded according to other aspects of the invention. Where motion vectors are available, motion vectors for the other pictures are used in interpolation and preferably motion vectors are communicated for pictures selected for communication but no motion vectors are communicated specifically for the interpolated pictures.

The adjustment information may include information concerning the timing of an interpolated frame to which the adjustment information relates. The adjustment information may include information concerning the motion between the interpolated frame and other communicated frames or residual frames. The adjustment information may include information describing global motion between the interpolated frame and other communicated frames or residual frames. The adjustment information may include information concerning the selection of parameters used for the control of, or modification of the operation of, the interpolation procedure. Adjustment information may be provided for all pictures to be interpolated. Some pictures may be interpolated without adjustment.

The invention extends to corresponding methods and apparatus for decoding.

The invention further provides a method of decoding a signal coded by a method according to anyone herein comprising reconstructing a portion of the signal to be reconstructed based on a received portion of data and a method of selective prediction of a subsequent portion corresponding to said method.

A method of decoding a signal coded according to the first aspect may comprise receiving and decoding first and second coded portions of data, selectively predicting a third portion of data based on the first and second portions and applying received coded data for the third portion to the selective prediction to reconstruct the third portion.

A method of decoding data coded hierarchically may comprise receiving data encoding a parent level and reconstructing the parent level and reconstructing the child level based on received data for the child level and re-using data used in the reconstruction of the parent level.

The invention extends to a computer program or computer program product for performing a method according to any method disclosed herein.

The invention further provides apparatus corresponding to all method aspects. The invention provides a coder arranged to perform any coding method. The invention provides a decoder arranged to perform any decoding method.

A general issue arises, relevant to the above coding scheme and others which enable compression of a video stream by storing data to enable at least some frames of the data stream to be predicted, at least partially, from other parts, typically preceding and subsequent frames (e.g. MPEG coding). It is not possible to access the compressed data stream directly at any random point in the data stream. Rather, it is necessary to find an access point in the data stream from which playback can commence, for example a compressed frame that is present in its entirety and from which other frames in the data stream may be accessed. An example of an access point may be an I-frame in an MPEG-compressed data stream and access points in the novel scheme discussed herein are explained, which may include points where statistical contexts are refreshed.

Thus a general problem is that it is difficult to provide random access to a compressed video stream, which means that it is difficult to enable a user to fast-forward through, or scrub, a coded data stream.

According to a further aspect, there is provided a method of processing a media data stream comprising:
providing a compressed media data stream having a plurality of access points, wherein the access points comprise points from which media playback can be commenced;
identifying access points in the media data stream;
inserting index data at a point in the media data stream to identify at least one subsequent access point.

The method may advantageously enable a media data stream to be compressed whilst enabling effectively random access to selected points in the data stream. This may provide greater flexibility to a user accessing the data stream, for example, by enabling a user to fast-forward through, or scrub, the data stream to a selected point, or rewind through the data stream. This may facilitate browsing and editing of the data stream.

In one embodiment, the step of providing a compressed media data stream may comprise receiving a compressed media data stream. Hence index data may be inserted into pre-compressed media data.

In an alternative embodiment, the step of providing a compressed media data stream may comprise receiving a media data stream and compressing the media data stream. Preferably, the steps of identifying access points and inserting index data are performed in conjunction with the step of compressing the media data stream. Hence index data may be inserted into the media data stream as the stream is being compressed.

In one embodiment, the media data stream comprises a video stream comprising a plurality of frames.

Preferably, the index data comprises an index table. Further preferably, each index table comprises an offset value indicating the position of the next access point or index data within the data stream. This may allow a decoder or browsing system to move directly between headers, access points or other index data in the data stream without parsing the intervening data stream to determine the location of each access point.

In one embodiment, each item of index data may comprise an indication of the structure of the frames immediately following the access point. For example, the index data may include information such as the number and location of P and B-frames in the subsequent data stream.

Preferably, the access points comprise elements in the data stream that have been compressed without reference to other elements in the data stream.

Preferably, the media data stream comprises a video stream and elements in the data stream comprise frames.

In one embodiment, the access points comprise I-frames in an MPEG-encoded data stream.

Preferred features of method aspects may be applied to apparatus and program aspects and vice versa. The coding methods of each aspect may be combined with each other (as in the embodiment) or combined with other coding methods, as may their preferred features.

Embodiments will now be described, by way of example, with reference to the accompanying drawings in which:-

Figure 1 shows overall hybrid encoder architecture;
Figure 2 shows rate-distortion curves for two signal components;
Figure 3 illustrates minimisation of the Lagrangian cost function;
Figure 4 illustrates frame coding architecture;
Figure 5 illustrates perfect reconstruction analysis and synthesis filter pairs;
Figure 6 illustrates wavelet transform frequency decomposition;
Figure 7 illustrates a 3-level wavelet transform of a library picture ("LENNA");
Figure 8 illustrates uniform and dead-zone quantisers, with mid point reconstruction values;
Figure 9 is an entropy coding block diagram;
Figure 10 illustrates a unary encoding tree;
Figure 11illustrates prediction of L1 and L2 frames when L1 frames are P frames;
Figure 12 illustrates prediction of L1 and L2 frames when L1 frames are also B-frames;
Figure 13 illustrates overlapping blocks in which the darker-shaded areas show overlapping areas;
Figure 14 illustrates sub-pixel motion-vector refinement;
Figure 15: shows the neighbouring vectors available in raster-scan order for local variance calculation;
Figure 16 illustrates Macroblock splitting modes;
Figure 17 illustrates motion vector entropy coding architecture;
Figure 18 demonstrates that data other than MB_SPLIT and MB_CBMODE is always associated with particular blocks, even if the relevant prediction unit is the sub-MB or MB itself;
Figure 19 illustrates how block data is scanned in raster order by MB and then in raster order within each MB;
Figure 20 shows that, for the purposes of prediction, values are deemed to be propagated within MBs or sub-MBs;
Figure 21 shows the aperture for a) block parameter prediction, and b) MB parameter prediction, showing shaded blocks repeated in the NWMP scheme;
Figure 22 shows how the MV prediction residue values are deemed to be propagated just as the values themselves are;
Figure 23 illustrates exp-Golomb VLC coding;

To assist in understanding the invention, a complete coding and decoding system will be explained. Certain components may be based on conventional components and so will not be described in detail, but it will be appreciated that the application in the context of this system may differ from conventional application. Although described in the context of a complete system for ease of understanding, it should be appreciated that the coder is essentially modular and, while there are certain synergies, features of the system may be provided independently of other features and in alternative combinations unless explicitly stated to be linked. As will become apparent, while certain components are novel per se, particularly in the adaptive prediction and interlace coding modules, other parts of the system comprise novel combinations of components which, while based on conventional components, combine in a novel manner to provide particular advantages and these combinations may be provided as building blocks for an alternative coder.

Overall, the codec is based on a conventional motion-compensated hybrid codec. The coder has the architecture shown in Figure 1 below, whilst the decoder performs the inverse operations.
There are four main elements or modules to the coder:

- Transform and scaling – this involves taking picture data and applying a transform (in this case the wavelet transform) and scaling and rounding the coefficients to perform quantisation;
- Entropy coding – this is applied to quantised transform coefficients and to motion vector (MV) data and performs lossless compression on them;
- Motion estimation (ME) – this involves finding matches for picture data from previously coded pictures, trading off accuracy with motion vector bit rate;
- Motion compensation (MC) – this involves using the motion vectors to predict the current frame, in such a way as to minimise the cost of encoding the residual data.

The following sections describe these modules in more detail, after first describing the rate-distortion framework used throughout the encoder.

Rate-Distortion Optimisation (RDO)

To make good decisions in compression it is desirable to be able to trade off the number of bits used to encode some part of the signal being compressed, with the error that is produced by using that number of bits (although a working coder can be made without using this principle). There is little point striving hard to compress one feature of the signal if the effect it produces is much more significant than compressing some other feature with fewer bits. In other words, it is desirable to distribute the bit rate to get the least possible distortion overall.

One basic explanation of how this can be done is the Principle of Equal Slopes, which states that the coding parameters should be selected so that the *rate of change* of distortion with respect to bit rate is the same for all parts of the system.

To explain why this is so, we consider two independent components of a signal. They might be different blocks in a video frame, or different sub bands in a wavelet transform. Compressing them at various rates using a selected coding technique tends to give curves like those shown in Figure 2. They show that at low rates, there is high distortion (or error) and at high rates there is low distortion, and there is generally a smooth curve between these points with a convex shape.

We assign B1 bits to component X and B2 bits to component Y and examine the slope of the rate-distortion curves at these points. At B1 the slope of X's distortion with respect to bit rate is much higher than the slope at B2, which measures the rate of change of Y's distortion with respect to bit rate. This is not the most efficient allocation of bits - consider increasing B1 by a small amount to B1+Δ and decreasing B2 to B2-Δ. Then the total distortion has reduced even though the total bit rate hasn't changed, due to the disproportionately greater drop in the distortion of X.

The conclusion is therefore that for a fixed total bit rate, the error or distortion is minimised by selecting bit rates for X and Y at which the rate-distortion curves have the same slope. Likewise, the problem can be reversed and for a fixed level of distortion, the total bit rate can be minimised by finding points with the same slope.

Two issues arise in practise, firstly, how to find points on these curves with the same slope and secondly, how to hit a fixed overall bit budget. The first question can be answered by Figure 3: the intercept of the tangent to the rate-distortion curve at the point (R0,D0) is the point R0+λD0 where –λ is the slope at the point (R0,D0). Furthermore it is the smallest value of R+λD for all values of (R,D) that lie on the curve. So in selecting, for example, a quantiser in a given block or sub band, one minimises the value D(Q)+λR(Q) over all quantisers Q, where D(Q) is the error produced by quantizing with Q and R(Q) is the rate implied.

In order to hit an overall bit budget (alternatively, distortion), one needs to iterate over values of the Lagrangian parameter λ in order to find the one that gives the right rate (alternatively, distortion). In practise, this iteration can be done in slow time given a reasonable encoding buffer size, and by modelling the overall rate distortion curve based on the recent history of the encoder.

Rate-distortion optimisation (RDO) is used throughout the video encoder of the embodiment and it has a very beneficial effect on performance (but is not essential to the invention). Although RDO is described mathematically, there is plenty of scope for a practical implementation to use empirical approximations, particularly to deal with the practical problems mentioned below.

1) There may be no common measure of distortion.
For example: quantising a high-frequency sub band is less visually objectionable than quantising a low-frequency sub band, in general. So there is no direct comparison with the significance of the distortion produced in one sub band with that produced in another. This can be overcome by perceptual weighting, in which the noise in HF bands is downgraded according to an estimate of the Contrast Sensitivity Function (CSF) of the human eye, and this is what is preferably done in an embodiment. The problem even occurs in block-based coders, however, since quantisation noise can be successfully masked in some areas but not in others. Perceptual correction factors are therefore desirable in RDO in all types of coders.

2) Rate and distortion may not be directly measurable.
In practice, measuring rate and distortion for, e.g. every possible quantiser in a coding block or sub band cannot mean actually encoding for every such quantiser and counting the bits and measuring MSE. However, what one can do is estimate the values using entropy calculations or assuming a statistical model and calculating, say, the variance. In this case, the R and D values may well be only roughly proportional to the true values, and an empirical factor(s) may be used to compensate for this effect in each component part of the encoder.

3) Components of the bit stream will be interdependent.
The model describes a situation where the different signals X and Y are fully independent. This is often not true in a hybrid video codec. For example, the rate at which reference frames are encoded affects how noisy the prediction from them will be, and so the quantisation in predicted frames depends on that in the reference frame. Even if elements of the bit stream are logically independent, perceptually they might not be. For example, with I-frame coding, each I-frame could be subject to RDO independently, but this might lead to objectionably large variations in quantisation noise between frames with low bit rates and rapidly changing content.

Frame coding

Frame coding will now be described. Both intra and inter frames are treated similarly in the codec. First they are wavelet-transformed using separable wavelet filters. Then they are quantised using RDO quantisers, before prediction (in the case of intra frames) and entropy coding.
The architecture of a frame coding implementation is shown in Figure 4.
As can be seen, each wavelet sub band is coded independently. This means that the encoding and decoding can be parallelised across all the sub bands.

Wavelet transform

The discrete wavelet transform is described in many sources, including J. Goswami and A. Chan, 'Fundamentals of Wavelets: Theory, Algorithms and Applications', Wiley 1999. In the codec it plays the same role of the DCT in MPEG-2, although applied to a whole picture rather than a block of data, in decorrelating data in a roughly frequency-sensitive way, whilst having the advantage of preserving fine details better. In one dimension it consists of the iterated application of a complementary pair of half-band filters followed by subsampling by a factor 2.

These filters are termed the analysis filters. Corresponding synthesis filters can undo the aliasing introduced by the critical sampling and perfectly reconstruct the input. This arrangement is illustrated in figure 5. Clearly not just any pair of half-band filters can do this, and there is an extensive mathematical theory of wavelet filter banks. The filters split the signal into a LH and HF part; the wavelet transform then iteratively decomposes the LF component to produce an octave-band decomposition of the signal.

Applied to two-dimensional images, wavelet filters are normally applied in both vertical and horizontal directions to produce four so-called sub bands termed Low-Low (LL), Low-High (LH), High-Low (HL) and High-High (HH). In the case of two dimensions, only the LL band is iteratively decomposed to obtain the decomposition of the two-dimensional spectrum shown in Fig. 6.

The number of samples in each resulting sub band is as implied by Figure 6: the critical sampling ensures that after each decomposition the resulting bands all have one quarter of the samples of the input signal.

The choice of wavelet filters has an impact on compression performance, filters having to have both compact impulse response in order to reduce ringing artefacts and other properties in order to represent smooth areas compactly. One criterion is the number of 'vanishing moments' of the high-pass filter. A filter has zeroth-order moments if it removes constant signals – any filter with a zero at DC does this; it has first-order moments if it removes linear signals; second-order moments if it removes quadratics; and so on. A filter has N vanishing moments if its z-transform has an N-order zero at unity. Suitable filters are the Daubechies (9,7) filter [M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, 'Image coding using wavelet transform', IEEE Trans. Image Processing, vo 1, pp 295-220, 1992] which if applied naively require an average of 8 multiplications per sample for the transform in both directions. However, the so-called 'lifting-scheme' [W. Sweldens, 'The lifting scheme: A custom-design construction of biorthogonal wavelets'. Technical Report 1994:7, Industrial Mathematics Initiative, Department of Mathematics, University of South Carolina, 1994] allows wavelet filters to be factorised and many implementations of these filters and similar exist using fewer operations (typical factorisations can reduce the number of multiplications by a factor of 3 or more).

Quantisation

Having transformed the frame, each sub band's coefficients are independently quantised using a so-called uniform dead-zone quantiser. A simple uniform quantiser is a division of the real line into equal-width bins, of size equal to the quantisation factor QF: the bins are numbered and a reconstruction value is selected for each bin. The bins consist of the intervals

$[(N\text{-}1/2)*QF, (N+1/2)*QF]$

for integers *N*, which are also the labels for the bin, and it is the *labels* that are subsequently encoded. The reconstruction value used in the decoder (and for local decoding in the encoder) can be any value in each of the bins. The obvious, but not necessarily the best, reconstruction value is the midpoint *N*QF*. See Figure 8a below.

A uniform dead-zone quantiser is slightly different in that the bin containing zero is twice as wide. So the bins consist of $[-QF,QF]$, with a reconstruction value of 0 and other bins of the form

$[N*QF, (N+1)*QF]$

for $N>0$ and

$[(N\text{-}1)*QF, N*QF]$

for $N<0$, with reconstruction points somewhere in the intervals. The bin structure is shown in Figure 8b with mid-point reconstruction points.

The advantage of the dead-zone quantiser is two-fold. Firstly, it applies more severe quantisation of the smallest coefficients, which acts as a simple but effective de-noising operation. Secondly, it admits a very simple and efficient implementation. The quantised value *N* of a coefficient *c* is given by the formula:

$$\begin{aligned} N &= \lfloor |c| / QF \rfloor \text{if c} \geq 0 \\ &= -\lfloor |c| / QF \rfloor \text{otherwise} \end{aligned}$$

where the braces $\lfloor\ \rfloor$ mean that the remainder is to be discarded. The corresponding reconstructed value $\tilde{c}$ is given by:

$$\begin{aligned} \tilde{c} &= 0 \,\text{if}\; N = 0 \\ &= (N + 0.375) * QF \;\text{if}\; N > 0 \\ &= (N - 0.375) * QF \;\text{if}\; N < 0 \end{aligned}$$

A value of 0.5, giving the mid-point of the interval might be the obvious reconstruction point, giving as it does the mid-point of the bin. Typically, however, the values of transformed coefficients in a wavelet sub band have a distribution with mean very near zero and which decays reasonably rapidly and uniformly for larger values (the Laplacian distribution, which has exponential decay, is often used to model wavelet coefficients). Values are therefore more likely to occur in the first half of a bin than in the second half and the smaller value of 0.375 selected reflects this bias, and gives better performance in practice. It is noted that a quantiser need not have uniformly distributed steps, and for a given set of data the optimum quantiser (in a rate-distortion sense) can be constructed by the so-called Lloyd-Max algorithm [R. Gray and D. Neuhoff, 'Quantisation', IEEE Trans. Info. Theory, vol. 44 no. 6, October 1998]. However, for data with a Laplacian distribution, this turns out to be a uniform quantiser.

Lagrangian parameter control of subband quantisation

The optimum quantiser is chosen for each sub band by computing the quantiser which minimises a Lagrangian combination of rate and distortion. Rate is measured via a zeroth-order entropy measure *Ent(q)* of the quantised symbols resulting from applying the quantisation factor *q*, calculated as a value of bits/pixel. Distortion is measured in terms of the perceptually-weighted mean-square-error, *MSE(q)*, resulting from the difference between the original and the quantised coefficients. Hence the total measure for each quantiser *q* is:

$$\frac{MSE(q)}{w} + \lambda \cdot Ent(q)$$

where *w* is the perceptual weight associated with the sub band – higher frequencies having a larger weighting factor. The quantisers are incremented in quarter-powers of 2 – i.e. *q* is an integer approximation of $2^{n/4}$ for integers *n*. In other words, the quantisers represent the coefficient magnitudes to variable fractional-bit accuracies in quarter-bit increments.

Suitable perceptual weighting may be based on that disclosed by A. Watson et al, 'Visual thresholds for wavelet quantisation error', SPIE Proc. Vol 2657, Hum. Vis. &Elec. Imag. B. Rogowitz and J. Allebach ed., 1996. In an implementation, for ease of use, the Lagrangian parameter λ may be entered as a user-selectable quantisation parameter. It may be convenient for the actual value to be entered in log units, since then there will be a roughly linear relationship with the resulting PSNR. The larger the value, the lower the resulting bit rate, and vice-versa.

Spatially-varying quantisation in wavelet coding

The codec can in principle apply a number of quantisers per sub band according to various schemes. We now describe another method for implicitly varying quantisation across a wavelet sub band according to a predefined spatial pattern. This principle of varying numbers of quantisers is not limited to wavelet compression but can be applied with other transforms, including without limitation block transforms such as DCT (MPEG, DV) or lapped orthogonal transforms; this feature may be independently provided.

In such a scheme, the quantiser *q(i,j)* applied to a coefficient *c(i,j)* at location *(i,j)* in a sub band may consist of the product of two values: the 'root' quantiser q and a weight value *w(i,j)* dependent on the coefficient position within the sub band –

$q(i,j)=q\cdot w(i,j)$

The weight function or matrix $w(i,j)$ may be derived from a look-up table available at both the decoder and the encoder, either by signalling the table to be used from a library of known tables or by signalling, at some earlier time, the entire table, or by some other method to ensure synchronisation.

Typically, such a function could be used to take advantage of the effects of overscan in the presentation of pictures as well as the tendency for viewers to be concerned with action at the centre of a picture. For this reason, a weighting matrix which increases quantisation levels at the edge of a subband may provide for bit-rate savings in areas which are either not seen or are not perceptually significant. This feature may be independently provided.

In this case the weighting matrix may (but need not) assume the same overall form across all sub bands, taking into account the relative sizes of those sub bands. For a given sub band B the weighting matrix w_B could thus derive from a 'master' weighting matrix W as follows:

$w_B(i,j)=K_B\cdot W(S_B\cdot i+X_B,\ S_B\cdot j+Y_B)$

where S_B is a scaling factor, X_B and Y_B are translation factors, and K_B is a multiplying factor, all specific to the sub band B. The invention is not restricted to deriving weighting matrices in this way, however.

Frame entropy coding

The entropy coding used in frame coding is based on four stages: coefficient prediction, binarisation, context modelling and adaptive arithmetic coding. All these stages are present for intra frame coding, but prediction is absent from inter-frame coding, since the picture has already largely been decorrelated by motion compensation.

The purpose of the first two stages is to provide a bit stream with easily analysable statistics that can be encoded using arithmetic coding which can adapt to those statistics to reflect local variations in picture characteristics. Suitable binarisation and adaptive context modelling may be based on the Context-Adaptive Binarized Arithmetic Coding (CABAC) used in various parts of the H.264/AVC standard [ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC, Draft standard, available at ftp://ftp.imtc-files.org/jvt-experts/], although implementation details may differ. The inventive adaptive prediction used herein, however, is quite different to conventional methods and will be described below in detail.

Prediction

The aim of the prediction stage is to remove any residual interdependencies between coefficients in the wavelet sub bands, so that subsequent entropy coding can be applied as far as possible to decorrelated data. Prediction only applies to intra frames.

Residual interdependencies within sub bands result in the main from edges in the original image. For example, a horizontal edge produces a horizontal line in the LH sub bands, which have been low-pass filtered horizontally and high-pass filtered vertically. In this case, it is possible to predict each coefficient $c(x,y)$ from its predecessor on the same line, $c(x\text{-}1,y)$. So the value to be entropy coded could instead be:

$d(x,y) = c(x,y) - c(x\text{-}1,y)$

i.e. the stream is equivalent to DPCM followed by entropy coding if the coefficients are scanned in raster order.

For HL bands, the roles of x and y would be swapped, since vertical edges will introduce vertical dependencies in these bands. HL bands should therefore be scanned by column rather than by row.

Prediction in diagonal (HH) bands could proceed by using diagonally adjacent coefficients. However, there is little practical benefit gained with a convenient diagonal predictor and hence embodiments (including the present embodiment) may have no prediction in HH bands. The only remaining band is the DC band. This band is a very low-resolution version of the original picture. In this band, the prediction operation is defined by:

$d(x,y) = c(x,y)\text{-}(\ c(x\text{-}1,y) + c(x,y\text{-}1)\ c(x\text{-}1,y\text{-}1))/3$

If the sub band is scanned in row (raster) or in column order then each of the predicting values is available to the decoder.

For pictures with low texture content, these prediction operations reduce bit-rate considerably. Unfortunately, for pictures with a large amount of texture, neighbouring coefficients are already decorrelated and the prediction actually increases bit rate. Most pictures are a mixture of these characteristics, and so the improvement is not clear-cut.

To deal with this problem, the embodiment makes the prediction adaptive, since edges and texture are local features. The approach taken is to predict if prediction would have worked for the previously scanned coefficient. For vertically-oriented sub bands (i.e. LH bands) the rule is:

$d(x,y) = c(x,y) - c(x\text{-}1,y)$ if $\alpha\left|c(x-1,y)-c(x-2,y)\right| < \left|c(x-1,y)\right|$

$\quad = c(x,y)$ otherwise

In one implementation, the best value of α was determined experimentally to be 6 (preferably at least 2, more preferably at least 3 and preferably also no more than 10, values of 4-8 being particularly suitable). This implies that the desired threshold for the efficacy of prediction is rather high. Adaptation is applied to LH and HL bands but not to the DC band.

This technique resulted in bit-rate reductions of up to 30% for some pictures, and increased the bit-rate of other, highly textured, pictures by less than 2% (which represents an acceptable increase for such a picture – in most practical sequences a significant decrease can be expected).

Adaptive prediction of wavelet coefficients

To start with some definitions, consider a stream of symbols $x(i)$. These could be binary symbols (0 or 1) or integers or any series of elements from a discrete alphabet on which it is possible to perform arithmetic operations. The symbol stream $x(i)$ could result, as in the case described above, from scanning coefficients from a wavelet sub band transform of an image in a predetermined order, but many other examples are possible, such as a stream of characters in a text document.

Denote by $S(i)$ the set of all symbols up to and including $x(i)$:
$S(i)=\{x(k) : k\leq i \}$

The symbols in $S(i\text{-}1)$ are all available for predicting $x(i)$. The result of predicting using prior symbols can be written:
$y(i)=x(i)-P(S(i\text{-}1))$
where P is some prediction operator. A traditional error-feedback predictor can adapt P on the basis of consideration of the values $y(k)$ and $x(i)$ for $k<i$ – that is, on the actual prior outputs of the predictor and on the prior inputs. This is illustrated in Figure 28.
In distinction to this approach, the invention conditions a predictor on the basis of what *would have been* the best prediction to have made for prior symbols. This distinction is illustrated in Figure 29.
The whole scheme can be thought of as deriving a new predictor P_1 from a simpler predictor P, by switching between a prediction of zero and that given by P.
In the example of a HL-filtered sub band of a wavelet transform of an image, each wavelet coefficient is predicted by its immediate predecessor on the same line. If the symbols are scanned in raster order (i.e. line by line) then the predictor P is simply given in this case by
$P(S(i))= x(i)$
since this is the previous symbol in scan order also. With these definitions, a new predictor P_1 can be defined by, for example, performing a simple weighted comparison between predicted and non-predicted outputs:

$$P_1(S(i-1))=\begin{cases} P(S(i-1))\,\text{if}\,\lambda\cdot|P(S(i-2))-x(i-1)|<|x(i-1)| \\ 0 \text{ otherwise}\end{cases} \qquad (*)$$

Generalisations

1) The adaptation mechanism can be combined with other means of adaptation, such as error feedback, and its component parts can themselves be adapted.
For example, the new predictor P_1 can be used as the predictor in an error-feedback scheme. By such means, all parts of the predictor – the subsidiary predictor P, the comparison means and the control means – can all be adapted. In particular, in the example given above, the weighting coefficient λ can be adapted. A great variety of signals are available for further adaptation:

a) The prior inputs to the prediction mechanism P_1;

b) The prior outputs of the prediction mechanism P_1;

c) The prior outputs of the subsidiary prediction mechanism P;

d) The prior outputs of the comparison mechanism.

Feedback elements can additionally be incorporated into the embodiment of the invention. In particular, it is possible to assess the past success of the new prediction mechanism P_1 and compare it with the success of using the subsidiary predictor P and use it to establish the reliability of the switch mechanism and so adjust either the control means or the comparison means.

2) The process can be iterated and the new prediction operator P_1 can be used as the subsidiary predictor to produce a further prediction operator P_2 by the same method.

3) Various methods can be used for performing the comparison and control means.

A simple generalisation of formula (*) would be to define

$$P_1(S(i-1)) = \begin{cases} P(S(i-1)) \text{ if } \lambda \cdot C(P(S(i-2)) - x(i-1)) < C(x(i-1)) \\ \quad 0 \text{ otherwise} \end{cases}$$

where C is any cost measure (such as an entropy measure) that may also be adapted over time, and λ is a parameter, that may also be adapted over time.

4) In the case of the coding a discrete signal in a compression system, quantisation may also be brought within the prediction loop.

In the example given of coding wavelet sub bands, the wavelet coefficients were assumed to have all been quantised prior to prediction and entropy coding. However it is possible to use prediction on unquantised data and then quantise the prediction residuals, provided that:

a) the coefficients used for prediction are reconstructed coefficients, on which inverse quantisation and prediction have been performed;

b) any adaptation is based only on reconstructed coefficients.

The adjustments required to the block diagram of Figure 29 are shown in Figure 30.

Binarisation

Binarisation is the process of transforming the multi-valued coefficient symbols into bits. The resulting bit stream can then be arithmetic coded. The reason for doing this is that the efficiency of arithmetic coding depends on the accuracy of the statistics that the encoder uses. Wavelet coefficients cover a wide range but are heavily clustered around zero (except in the LL band). So the symbol space is large but most possible symbol values occur rarely or not at all. Maintaining a probability model of all the possible values that could occur in a quantised sub band consumes a lot of memory, and results in inaccurate estimates of probabilities since the sample space for most symbols is so small. Where the arithmetic coding is adaptive, as it is here, this can impact on coding efficiency significantly.

The simplest way to binarize a symbol is directly: a symbol is encoded by encoding the constituent bits of the binary representation of its magnitude, followed by a sign bit. This is termed bit-plane coding. Modelling the resulting bit stream in order to code it efficiently is possible and has been done but somewhat complicated. Each bit-plane has different statistics, and needs to be modelled separately taking into account interdependencies between bit-planes.

Modelling many conditional probabilities is possible if complex, of course, but in practice tends to reintroduce the problem of small sample sizes.

Instead, the arithmetic coding scheme used in the codec of the embodiment binarizes coefficient magnitudes with so-called unary coding, subsequently encoding sign bits for non-zero magnitudes. Unary coding is a simple VLC in which every non-negative number *N* is mapped to *N* zeroes followed by a 1 as shown in Fig. 10.

Unary encoding is actually optimal for a Laplacian probability distribution where the probability of *N* occurring is $2^{-(1+|N|)}$ (This is because the length of each symbol is bits, which is precisely the optimal length $-\log_2(p)$ predicted by information theory, where p is the probability of occurrence). A different context, or set of contexts, can be defined for each bin or for a number of bins together, and the effect is to analyse the coefficient statistics in an iterative way. For example, the probability of zero occurring in Bin 1 is the probability that the coefficient magnitude is >0, whilst *P*(0 : Bin 2) is the probability that the magnitude is >1, and so on. The sparse statistics of the larger coefficients can be effectively combined by having a single context for all the larger bins, representing the tails of the distribution. The process will be further explained by example.

For example, we suppose one wished to encode the sequence:
-3 0 1 0 -1
When binarized, the sequence to be encoded is:
0 0 0 1 | 0 | 1 | 0 1 | 1 | 1 | 0 1 | 0
The first 4 bits encode the magnitude, 3. The first bit is encoded using the statistics for Bin1, the second using those for Bin 2 and so on. When a 1 is detected, the magnitude is decoded and a sign bit is expected. This is encoded using the sign context statistics; here it is 0 to signify a negative sign. The next bit must be a magnitude bit and is encoded using the Bin 1 contexts; since it is 1 the value is 0 and there is no need for a subsequent sign bit. The same principle is applied iteratively.

Context modelling

Suitable context modelling of this embodiment is based on the principle that small coefficients, particularly zero, are well-predicted by their neighbours whilst large coefficients are not. Therefore the codec conditions the probabilities used by the arithmetic coder for coding bins 1 and 2 on the size of the neighbouring coefficients.

The reason for this approach is that, whereas prediction removes correlation between a coefficient and its neighbours, they may not be statistically independent even if they are uncorrelated: small and especially zero coefficients in wavelet sub bands tend to clump together, located at points corresponding to smooth picture areas.

To compute the context, a value *nhood_sum* is calculated at each point (*x,y*) of each sub band, as:

$$nhood\_sum(x,y) = |c(x-1,y)| + |c(x,y-1)|$$

(NB: *nhood_sum* depends on the size of the neighbouring coefficients, not on the *predicted* neighbouring coefficients *d*(*x,y*) as defined above).

There are nine contexts used, which are as follows:

1. FRAME_SIGN_CTX – context for sign bits;
2. FRAME_BIN1a_CTX – context for Bin 1, 0<*nhood_sum*<3
3. FRAME_BIN1b_CTX – context for Bin 1, *nhood_sum* ≥ 3
4. FRAME_BIN1z_CTX – context for Bin 1, *nhood_sum*=0
5. FRAME_BIN2a_CTX – context for Bin 2, *nhood_sum*<3
6. FRAME_BIN2b_CTX – context for Bin 2, *nhood_sum* ≥ 3
7. FRAME_BIN3_CTX – context for Bin 3
8. FRAME_BIN4_CTX – context for Bin 4
9. FRAME_BIN5plus_CTX – context for Bin 5 or more

After binarization, a context is selected based on the value of *nhood_sum* and on the bin, and the probabilities for 0 and 1 that are maintained in the appropriate context will be fed to the arithmetic coding function along with the value itself to be coded.

In the example of the previous section, when coding the first value, –3, the encoder then checks the values of neighbouring coefficients and produces the value *nhood_sum*. Based on this value, a different statistical model (that is, a count of 1 and a count of zero) is used to code the first two bins. So the coder maintains, for example, the probabilities that Bin 1 is 0 or 1, *given that* the value of neighbouring coefficients is 0 – this is contained in FRAME_BIN1z_CTX. These are fed to the arithmetic coding engine for encoding the bit in Bin 1, and the context probabilities are updated after encoding.

Arithmetic coding

Examples of Arithmetic coding are known, for example as disclosed in A. Moffat et al, 'Arithmetic coding revisited', ACM Transactions on Information Systems, 16(3), pp256-294, July 1998 so it will only be described briefly. Conceptually, an arithmetic coder can be thought of a progressive way of producing variable-length codes for entire sequences of symbols based on the probabilities of their constituent symbols. For example, if we know the probability of 0 and 1 in a binary sequence, we also know the probability of the sequence itself occurring. So if

$P(0)=0.2,\ P(1)=0.8$

then

$P(11101111111011110101)=(0.2)^3(0.8)^{17}=1.8\text{x}10^{-4}$ (assuming independent occurrences).

Information theory then says that optimal entropy coding of this sequence requires $\log_2(1/P)$=12.4 bits. Arithmetic coding produces a code-word very close to this optimal length, and implementations can do so progressively, outputting bits when possible as more arrive. Using a look-up table VLC for 20-bit sequences is, by contrast, less practical, requiring several Mbytes storage. VLCs for long sequences do exist, and depend on formulae for generating codewords on the fly. An example is the exp-Golomb codes used for the DC coefficient coding in Intra frames, described below. Adaptive VLCs also exist, based on a parameterised statistical model. In both cases, efficiency is only very high if the model used is accurate. Arithmetic coding can adapt to any statistics.

All arithmetic coding (AC) requires is estimates of the probabilities of symbols as they occur, and this is where context modelling fits in. Since AC can, in effect, assign a fractional number of bits to a symbol, it is very efficient for coding symbols with probabilities very close to 1, without the additional complication of run-length coding. The aim of context modelling is, in fact to use information about the symbol stream to be encoded to produce accurate probabilities as close to 1 as possible. This is illustrated by example, considering a bit stream with P(0)=0.5=P(1). If we knew that the bit stream was actually composed of two bit streams (perhaps alternated), in one of which P(0)=0.25 and P(1)=0.75 and in the other P(1)=0.25 and P(0)=0.75 then we could feed these more refined probabilities to the arithmetic coder as appropriate. The result would be a bit rate saving of 19%.

The estimates are computed for each context simply by counting their occurrences. In order for the decoder to be in the same state as the encoder, these statistics cannot be updated until after a binary symbol has been encoded. This means that the contexts must be initialised with a count for both 0 and 1, which is used for encoding the first symbol in that context.

An additional source of redundancy lies in the local nature of the statistics. If the contexts are not refreshed periodically then later data has less influence in shaping the statistics than earlier data, resulting in bias, and local statistics are not exploited. A simple way to refresh the contexts is to pick a number, preferably at least about 1000, preferably no more than about 10,000. (8092 is used in the codec) so that if the count of 0 plus the count of 1 for that context exceeds that number then these counts are halved. The effect is to maintain the probabilities to a reasonable level of accuracy, but to keep the influence of all coefficients roughly constant.

Alternative Binarisation Schemes

One embodiment may use adaptive binary arithmetic coding to achieve lossless entropy coding. In binary coding the only symbols encoded are 0 and 1. Thus multi-valued symbols must be converted to a string of 1's and 0's. The expedient of simply coding the binary representation of a multi-valued symbol is ineffective. The same embodiment may use "unary" coding for binarisation of prediction residue wavelet coefficients. It uses exp-Golomb (see below) coding for some other parameters.

Unary coding is optimal for a Laplacian distribution of values where the probability density of value x is 1/2(|x|+1). This should be a good approximation for prediction residue wavelet coefficients. However some parts of the picture, e.g. intra coded blocks, may deviate substantially from a Laplacian distribution. For these regions an alternative binarisation might improve compression efficiency. An exp-Golomb code is suitable for this. It is basically an exponent-mantissa representation where the exponent is coded using unary coding.

One possible implementation is to switch to alternative binarisation (e.g. exp-Golomb) when parent residual wavelet coefficients are greater than a threshold. The embodiment described above uses a single probability (context) when wavelet coefficients are greater than 4. This might be an appropriate threshold at which to switch to an alternative binarisation. This scheme uses implicit signalling (i.e. is parent coefficient > threshold) to change binarisation and so does not incur additional overhead to signal the switch. An alternative, similar to using multiple quantisers within a sub-band, may be to use alternative binarisations in different regions of the picture.

The use of alternative binarisations would require a new context structure with corresponding significant knockons to the rest of the codec design. The conceptual complexity of these changes is small. We already have the concepts of changes to the quantiser (and associated contexts) when the parent is non-zero and the embodiment already uses the exp-Golomb binarisations. The potential gains from such an approach are expected to be of the same order of magnitude as the use of multiple quantisers within a sub-band, i.e. on the order of 1%. The possibility of alternative binarisations may be flagged, in a coding scheme. One possibility is to flag this in the version number (a higher version number indicating this capability).

Non-linear Quantisers

In one embodiment the wavelet transform of the prediction residuals are quantised using uniform dead zone quantisers. That is a linear quantiser except that the zero bin is twice the width of the non-zero bins. The quantisers are parameterised by the size of the bins. Different quantisers are used for each transform sub-band.

Ideally the quantiser should match the probability density function of the magnitude of the signal to be quantised. For audio compression non-linear quantisers have proved beneficial and non-linear quantisers may be used in the invention. However, in many cases the probability distribution function of the transformed residuals is likely to approximate a Laplacian distribution and the optimum quantiser for such a distribution is a linear quantiser. Thus, the advantages of using (or providing the option of) non-linear quantisers is likely to be small but this may be tried in some applications.

Motion estimation and motion compensation

Motion estimation will now be described.

GOP structures

Three types of frames are defined in the codec. Intra frames (I frames) are coded without reference to other frames in the sequence, whereas inter frames are predicted by motion estimation from other frames. Level 1 frames (L1 frames) are inter frames which are also used as temporal references for other frames. The embodiment provides for L1 frames to be either P frames (i.e. only forward predicted), or B frames (bi-directionally predicted). Level 2 frames (L2 frames) are conventional B frames which are not used as a temporal reference. It is possible in either case to have more than one reference and indeed in the embodiment the references for P/L1 frames are (in temporal order) the previous I frame and the preceding L1 frame, if one exists (see Figure 11).

A practical embodiment may allow L1 frames to be B-frames. This might be useful in some applications, such as streaming, perhaps, where the key issue might not be the overall latency (which must always be at least one GOP) but the frequency of Intra frame updates. The references in this case are the (temporally) prior L1 frame and the succeeding I-frame, or the preceding and succeeding I-frame, if the first L1 frame is being coded (Figure 12). In certain cases, the second configuration may confer advantages.

Multiple Reference Frames

A basic implementation of the codec implements an MPEG style GOP structure. A more flexible structure is possible and would result in a more flexible codec with potentially better compression performance.

The basic codec uses level 0 (I frames), level 1 and level 2 pictures. These are roughly equivalent to the I, P and B frames used in MPEG coding. Level 0 frames are independently coded (intra) pictures. Level 1 frames are predicted (from 1 or 2 reference frames) and are, themselves, used as reference frames. Level 2 frames are predicted, but are not used as reference frames. In a basic implementation, level 2 frames are always predicted from 2 reference frames but it may be better if they could, optionally, be predicted from only a single frame.

More flexibility can be attained by maintaining a buffer of reference frames and allowing frames to be predicted from 1 or 2 of the reference frames. This would relax the requirement for a fixed GOP structure and thereby allow frames to be predicted in an adaptive and optimum way. It would allow, for example, the backward or bi-directional prediction of level 1 (P) frames from (future) I frames. It would also allow the use of long-term reference pictures that might, for example, contain the background of a scene; this feature may be independently provided. Long-term references might also be useful in programmes that frequently switched between two viewpoints, e.g. interviews.

The question arises as to how many frames should there be in the reference frame buffer. To implement MPEG style GOPs 3 reference frames are required (an I frame and 2 P frames). A buffer size of 4 frames represents only a small increase but gives much greater flexibility. This gives the 3 frames currently required plus 1 extra, which might be used as either a second I frame (for bi-directional prediction of level 1 frames) or for a long-term reference picture. The size of the buffer chosen in a given implementation depends on requirements. One option is to allow it to be defined in a coder profile. However, one objective is to minimise complexity and fixing decisions, where this can reasonably be done, reduces complexity. It could be argued that a large buffer would give more flexibility. Even so 4 frames is sufficient and limits the memory requirements of the decoder and is preferred.

It can also be argued that allowing prediction from more than 2 frames would be beneficial. Nevertheless 2 frames is found to be practically sufficient and restricting it to this number helps to minimise decoder complexity.

Adopting a more general prediction structure has a number of consequences. Firstly, the process of re-ordering for display is separated from that of prediction. Reference frames may have been displayed a long time ago, or may never be displayed. It will therefore be necessary to include signalling to indicate when a particular frame in the decoded picture buffer should be displayed. Secondly, I frames do not necessarily indicate random access points any more, since an L1-frame could be predicted by a reference frame (such as a long-term reference) prior to the preceding I-frame. So random access points also need signalling.

These enhancements require restructuring the decoder and the encoder. However the existing structure could be used to decode sequences with an appropriately restricted set of allowed reference frames.

To summarise the advantageous features:-
A coder may use a buffer of reference frames, preferably containing at least 4, preferably exactly 4 frames, rather than a fixed GOP structure.
A coder may allow the use of long-term reference frames.
A coder may permit the use of reference frames that are not displayed.
Level 2 pictures may, optionally, be predicted from either 1 or 2 reference frames (rather than always from 2 frames as in the current implementation).
The reference frame buffer size should be 4 frames.
The number of reference frames used to predict a frame should be limited to 2.

<u>Overlapped Block-based Motion Compensation</u>

Motion compensation in the codec uses Overlapped Block-based Motion Compensation (OBMC) to avoid block-edge artefacts which would be expensive to code using wavelets; this is a known technique, disclosed in G. Heising, D. Marpe, H. Cycon and A. Petukhov, 'Wavelet-based very low bit-rate video coding using image warping and overlapped block motion compensation', IEE Proceedings - Vision, Image and Signal Processing, vol. 148, no. 2, pp. 93-101, April 2001. The size of blocks can be varied with a desired degree of overlap selected: this is configurable within the codec. Although the codec need not be designed to be scalable, in the embodiment the size of blocks is the only non-scalable feature, and for lower resolution pictures, smaller blocks can easily be selected.

The OBMC scheme adopted in the embodiment is based on a separable Raised-Cosine mask. This acts as a weight function on the predicting block. Given a pixel $p=p(x,y,t)$ in frame t, p may fall within only one block or in up to four if it lies at the corner of a block (see Figure 13).

Each block that the pixel p is part of has a predicting block within the reference frame selected by motion estimation. The predictor $\tilde{p}$ for p is the weighted sum of all the corresponding pixels in the predicting blocks in frame t', given by $p(x-V_i, y-W_i, t')$ for motion vectors (V_i, W_i). The Raised-Cosine mask has the necessary property that that sum of the weights will always be 1:

$$\tilde{p}(x,y,t)=\sum_i w_i p(x-V_i, y-W_i, t'),\ \sum_i w_i = 1$$

Although this may seem *prima facie* complicated, in a practical implementation the additional complexity is to apply the weighting mask to a predicting block before subtracting it from the picture and this is feasible.

Motion estimation

Motion estimation will now be described, beginning with Hierarchical motion estimation.

Hierarchical motion estimation

The embodiment of the codec supports motion estimation to 1/8 pixel accuracy. Motion estimation (ME) is by far the single largest computational load in encoding and in many cases it is impractical to perform brute-force motion vector searches over reasonable areas, especially for HD material. Various shortcuts are desirable and the technique adopted herein is a Rate-Distortion Optimised hierarchical search method.

In this method, integral pixel accuracy is produced by hierarchical ME, and then refined to sub-pixel accuracy. The hierarchical approach repeatedly downconverts both the current and the reference frame by a factor of two in both dimensions, four times in all. Motion vectors are estimated for blocks at each level and used as a guide for the next higher resolution. The block size remains constant (and the blocks will still overlap at all resolutions) so that at each level there are only a quarter as many blocks and each block has 4 children at the next higher resolution. The lower-resolution block's motion vector is then used as a guide vector to be refined by the children at the next highest level of resolution. At each resolution, block matching proceeds by searching in a small range around the guide vector for the best match using the RDO metric (which is described below).

Downconversion and search ranges in hierarchical ME

The hierarchical approach dramatically reduces the computational effort involved in motion estimation for an equivalent search range. However it has two significant risks that need to be mitigated in an implementation.

The first risk is that when there is a variety of motions in a sequence, the hierarchical method matches the motion to that of the largest objects at an early stage and it is impossible to then escape to better matches at later stages because the search ranges are too small. To mitigate this, the codec does two things: it always searches around the zero vector (0,0) as well as around the guide vector – this allows it to track fast and slow-moving objects; secondly, it always searches a range of +/-2 pixels at each level – this means that an error of 1 pixel at the next lowest resolution can always be corrected.

The second risk is less apparent, but we have found it can still be a problem with very highly textured sequences. This is that the aliasing resulting from repeated downconversion with short (e.g. 1/2, 1/2) filters renders motion estimation difficult at lower levels, and can produce poor guide vectors. This is remedied in the codec by using the non-separable filter:

$$\begin{array}{cccc} & 1 & 1 & \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ & 1 & 1 & \end{array}$$

which reduces aliasing.

Variable Levels of Wavelet Decomposition

One embodiment may use a set number, here exactly 4 levels, of sub-band decomposition. However, the invention is applicable to a wide range of applications from low resolution streaming video to HDTV. The optimum number of levels of wavelet decomposition may vary with the application.

It is proposed to be able to vary the number of levels of wavelet decomposition. The number of levels is preferably signalled once per frame. The use of variable levels of wavelet decomposition may improve flexibility, scalability and improve compression efficiency for some image formats. However, there are interdependencies between motion compensation block sizes, colour resolutions and the number of levels used and so not all image formats will benefit from the ability to use an arbitrary number of levels of decomposition.

The data overhead from a variable number of levels of wavelet decomposition is negligible. A single bit per frame could be used to signal “same as before”. Although of course the implementation is different if a variable number of levels is allowed, the software complexity will not be significantly increased and computational requirements are largely unaffected.

Sub-pixel refinement and upconversion

Sub-pixel refinement also operates hierarchically. Once pixel-accurate motion vectors have been determined, the reference picture is upconverted using a Nyquist windowed-sinc function filter. Each block has an associated vector (V_0,W_0) where V_0 and W_0 are multiples of 8 with respect to the *upconverted* reference. 1/2-pel accurate vectors are found by finding the best match out of (V_0,W_0) and its 8 neighbours: (V_0+4,W_0+4), (V_0,W_0+4), (V_0-4,W_0+4), (V_0+4,W_0), (V_0-4,W_0), (V_0+4,W_0-4), (V_0,W_0-4), (V_0-4,W_0-4). This in turn produces a new best vector (V_1,W_1), which provides a guide for 1/4-pel refinement, and so on. The process is illustrated in Figure 14.

RDO motion estimation metric

The performance of motion-estimation and motion-vector coding is very important critical to the performance of a practical video coding scheme, as motion vectors can often comprise the majority of the information used in the encoding of inter frames, particularly B- frames. With motion vectors at 1/4 or 1/8th pixel accuracy, a simple-minded strategy of finding the best match between frames can greatly inflate the resulting bit rate for little or no gain in quality. What is required is the ability to trade off the vector bit rate with prediction accuracy and hence the bit rate required to code the residual picture and the eventual quality of that picture.

A simple but effective way we have found to do this is to incorporate a smoothing factor into the metric used for matching blocks. So the metric consists of a basic block matching metric, plus some constant times a measure of the local motion vector smoothness.

The basic block matching metric used is Sum of Absolute Differences with DC removal (DCSAD). Given two blocks X,Y of samples, this is given by:

$$DCSAD(X,Y) = \sum_{i,j} \left| x_{i,j} - y_{i,j} - DC(X,Y) \right|$$

where $DC(X,Y) = \frac{1}{N} \sum_{i,j} x_{i,j} - y_{i,j}$ is the average of the differences.

DC-removal helps maintain motion vector accuracy in fades and lighting changes.

The smoothness measure used is the local variance between the selected motion vector and previously computed motion vectors. So if the blocks are estimated in raster-scan order then vectors for blocks to the left and above are available for calculating the local variance, as illustrated in Fig. 15.

The local variance is defined, if the vectors of the neighbouring blocks are $\mathbf{V}_i$, by the recipe:

$$VAR(\mathbf{V}) = \sum_{i \in N} \left\| \mathbf{V} - \mathbf{V}_i \right\|^2, \quad \text{where } \left\| \mathbf{W} \right\|^2 = \mathbf{W} \cdot \mathbf{W} \quad \text{(inner product)}$$

The total metric is a combination of these two metrics. Given a vector **V** which maps the current picture block *X* to a block *Y*=**V**(*X*) in the reference frame, the metric is given by:

$$M(X,Y) = DCSAD(X,Y) + \lambda \cdot VAR(\mathbf{V})$$

The value λ is a coding parameter used to control the trade-off between the smoothness of the motion vector field and the accuracy of the match. When λ is very large, the local variance dominates the calculation and the motion vector which gives the smallest metric is simply that which is closest to its neighbours. When λ is very small, the metric is dominated by the SAD term, and so the best vector will simply be that which gives the best match for that block. For values in between, varying degrees of smoothness can be achieved.

The parameter λ can be set at the command-line of the encoder, but if omitted is calculated as a multiple (currently 0.1) of the Lagrangian rate-control parameters for the L1 and L2 frames (see above), so that if the inter frames are compressed more heavily then smoother motion vector fields are derived.

Macroblock structures and motion vector data

This section describes the Macroblock structures which are used to introduce a degree of adaptation into motion estimation by allowing the size of the blocks used to vary. The motion estimation stage of the encoding is organised by macroblock, and each combination of block size and prediction mode is tried using the RDO block-matching metric, and the best solution adopted macroblock by macroblock.

Macroblocks and variable-sized block matching

A macroblock consists of a 4x4 array of blocks, and there are three possible ways of splitting a MB, which are encoded in the MB variable MM_SPLIT:

MB_SPLIT=0: no split, a single MV per reference frame for the MB;

MB_SPLIT=1: split into four sub-macroblocks (sub-MBs), each a 2x2 array of blocks, one MV per reference frame per sub-MB;

MB_SPLIT=2: split into the 16 constituent blocks.

It is noted that, while still somewhat complex, the structures defined herein are considerably simpler than the 7 possible MB splittings, together with prediction from any two of up to 5 reference frames, and several special forms of motion vector prediction conventionally proposed in standard H.264.

The splitting mode is chosen by redoing motion estimation for the sub-MBs and the MB as a whole, again using the RDO metric described in the previous section, suitably scaled to take into account the different sizes of the blocks. At the same time, the best prediction mode (PREDMODE) is chosen. Four prediction modes are available for each prediction unit (block, sub-MB or MB):

MODE_NO_PRED: no prediction – intra coded;

MODE_REF1_ONLY: only predict from the first reference;

MODE_REF2_ONLY: only predict from the second reference (if one exists);

MODE_REF1AND2: bi-directional prediction.

So a different prediction mode can be chosen for each prediction unit of the MB. However, mode data itself incurs a cost in bit-rate. So a further MB parameter is defined, MB_CBMODE, which records whether a common block prediction mode is to be used for the MB:
MB_CBMODE=1: the same prediction mode is to be used for each prediction unit with the MB;
MB_CBMODE=0: the prediction modes of each block or sub-MB are different.

For example, if MB_SPLIT=1 then the MB is split into 4 sub-MBs, each itself combining four basic blocks. If MB_CBMODE=1 then if one of those sub-MBs has PREDMODE equal to MODE_REF1_ONLY, then *all* the sub-MBs have this prediction mode. Of course if MB_SPLIT is 0, then this is true by default – in this case MB_CBMODE need not be coded.

The result is a hierarchy of parameters: MB_SPLIT determines whether MB_CBMODE needs to be coded for a given macroblock; the MB parameters together determine which prediction units need to encode PREDMODE; and PREDMODE itself determines what motion vectors need to be present.

In motion estimation, an overall cost for each MB is computed, and compared for each legal combination of MB_SPLIT, MB_CBMODE and value of PREDMODE. Then the best combination is selected for coding.

<u>Block data</u>

Parameters other than MB_SPLIT and MB_CBMODE are termed block data, even though they may apply to blocks, sub-MBs or the MB itself depending on the value of the MB data. PREDMODE has already been described. The four remaining block parameters are:
REF1_x: horizontal component of motion vector to the first reference frame;
REF1_y: vertical component of motion vector to the first reference frame;
REF2_x: horizontal component of motion vector to the second reference frame;
REF2_y: vertical component of motion vector to the second reference frame.
Clearly not all of these values must be coded. If PREDMODE is MODE_REF1_ONLY then REF2_x and REF2_y will not be coded, for example.

<u>Motion vector coding</u>

Motion vector (MV) coding is important to the performance of video coding, especially for codecs with a high level of MV accuracy (1/4 or 1/8 pel). For this reason, MV coding and decoding is probably the most complicated part of the wavelet codec since significant gains in efficiency can be made by choosing a good prediction and entropy coding structure. The basic format of the MV coding module is similar to the coding of quantised coefficients: it consists of prediction, followed by binarisation, context modelling and adaptive arithmetic coding (Figure 17).

Median prediction of motion vector data

Some of the conventions used herein will first be explained.

All the motion vector data is predicted from previously encoded data from nearest neighbours using a form of median prediction, which is described below. The two parameters MB_SPLIT and MB_CBMODE are deemed to comprise the MB data –all other parameters, even though (depending on the MB parameters) they may refer to blocks, sub-MBs or MBs, are deemed to be *associated* with particular blocks, and are termed block data. This allows for a consistent prediction and coding structure to be adopted.

Example. If MB_SPLIT=1 and MB_CBMODE=0 then the prediction units in a MB are sub-MBs. Nevertheless, the prediction mode and any motion vectors are associated with the top-left *block* of each sub-MB and values need not be coded for other blocks in the sub-MB.

Example. If MB_SPLIT=2 but MB_CBMODE=1 then the block parameter PREDMODE need only be coded for the top-left block in the MB. Motion vectors need to be coded for every block in the MB if PREDMODE is not equal to MODE_NO_PRED.

The second convention is that all MB data is scanned in raster order for encoding purposes. All block data is scanned first by MB in raster order, and then in raster order within each MB. That is, taking each MB in raster order, each block value which needs to be coded within that MB is coded in raster order (see Figure 19).

The third convention concerns the availability of values for prediction purposes when they may not be coded for every block. Since prediction will be based on neighbouring values, it is necessary to propagate values for the purposes of prediction when the MB modes have conspired to ensure that values are not required for every block.

Example. In Figure 20 below, we can see the effect of this. Suppose we are coding REF1_x. In the first MB, MB_SPLIT=0 and so at most only the top-left block needs a value, which can be predicted from values in previously coded MBs. As it happens, PREDMODE=MODE_REF1_ONLY and so a value is coded. The value *v* is then deemed to be applied to every block in the MB. In the next MB, MB_SPLIT=1 and MB_CBMODE=0, so the unit of prediction is the sub-MB. In the top-left sub-MB PREDMODE is, say, MODE_REF1AND2 and so a value *x* is coded for the top-left block of that sub-MB. It can be predicted from any available values in neighbouring blocks, and in particular the value *v* is available from the adjacent block.

Neighbour-weighted median prediction

Before entropy coding the MV data is predicted in order to remove correlation, just as in Intra frame coding of wavelet subbands. In this case the prediction used is what I term *neighbour-weighted median prediction* (NWMP). This is median prediction from previously-coded neighbouring blocks/MBs, but with a bias towards those neighbours closest to the block being predicted. The aperture for the NWMP predictor is shown in Figure 21 below.

In many cases values are not available from all blocks in the aperture, for example if the prediction mode is different. In this case the blocks are merely excluded from consideration.

NWMP works by taking values associated to the nearest blocks, shown shaded in the diagram, and repeating them twice in the list of values from which the median is taken. The purpose of NWMP is that, in the event (for example) of a prediction mode change between a block and its neighbours, no predictor may be available from the nearest neighbours and so more distant neighbours should be consulted. On the other hand, if the nearest neighbours' values are available then they should be given more weight. Hence a larger aperture with neighbour-weighting. NWMP is, however, complex, and empirical attempts to simplify it, while permissible in implementations, should consider carefully the effects on performance.

In the case of the MB data, the number of possible values is only 3 in the case of MB_SPLIT and 2 in the case of MB_CBMODE. The prediction therefore can use modulo arithmetic and produces an unsigned prediction residue of 0,1 or 2 in the first case and 0 or 1 in the second. All other predictions produce signed prediction residues. There are also only four prediction modes, so prediction modulo 4 could be used for PREDMODE; however, this number could increase if more potential reference frames are allowed. Median prediction for MB data and for PREDMODE may not seem *prima facie* intrinsically sensible as there is no concept of order. However, in practice it actually works reasonably well, and this data comprises a small proportion of the MV data as a whole, so a unified approach has been maintained.

Entropy coding

Entropy coding of the MV prediction residuals uses the same basic architecture as for wavelet coefficient coding: unary VLC binarization, followed by adaptive arithmetic coding with multiple context models. For MV coding there are many different types of data, and these have their own context models.

There are 24 motion vector data contexts in total. However, there are many components of the MV data, and the majority are not in play at any one time. The contexts for MB_SPLIT residues are:
MB_CTX_SPLIT_BIN1: context for Bin 1 of the unary binarization;
MB_CTX_SPLIT_BIN 2: context for Bin 2;
MB_CTX_SPLIT_BIN 3: context for Bin 3.
There is only one context required for MB_CBMODE, since the value of residues is either 0 or 1:
MB_CTX_CBMODE
PREDMODE residues are more numerous – there are five:
MODE_CTX_BIN1: context for Bin 1 of magnitude the bits;
MODE_CTX_BIN2: context for Bin 2 of magnitude the bits;
MODE_CTX_BIN3: context for Bin 3 of magnitude the bits;

MODE_CTX_BIN4plus: context for the remaining bins of the magnitude bits;
MODE_CTX_SIGN: context for sign bits.

The remaining data is the actual motion vector values: REF1_x, REF1_y, REF2_x, REF2_y. As for wavelet coefficient coding, these values are contextualised on the basis of the size of their neighbours, although in this case it is the size of the neighbouring prediction residuals not the neighbouring values themselves. For the purposes of context modelling, residue values are assumed to be propagated in just the same way that the values themselves are – so after prediction, the propagated values corresponding to Figure 20 are illustrated in Figure 22 below.

If the neighbouring residues $r(x\text{-}1,y)$ and $r(x,y\text{-}1)$ are available then a value *nhood_residue_size* can be defined by:

$$nhood\_residue\_size(x,y)=|r(x-1,y)|+|r(x,y-1)|$$

If only one of these values is available, then *nhood_residue_size* is equal to twice its magnitude. Only Bin 1 is contextualised on *nhood_residue_size* – the contexts for horizontal vector components are (vertical components are similar):

VEC_CTX_X_BIN1a: Bin 1 context for magnitude bits, *nhood_residue_size*<3;
VEC_CTX_X_BIN1b: Bin 1 context for magnitude bits, 3≤*nhood_residue_size*≤15;
VEC_CTX_X_BIN1c: Bin 1 context for magnitude bits, *nhood_residue_size*>15 or $r(x\text{-}1,y)$ and $r(x,y\text{-}1)$ are not available.

The remaining horizontal vector contexts are:
VEC_CTX_X_BIN2: Bin 2 context for magnitude bits;
VEC_CTX_X_BIN3: Bin 3 context for magnitude bits;
VEC_CTX_X_BIN4: Bin 4 context for magnitude bits;
VEC_CTX_X_BIN5plus: context for the remaining bits of the magnitude bits;
VEC_CTX_X_SIGN: context for the sign bits.

The resulting compression is quite powerful, but rather complex; ideally, the number of contexts should be reduced.

Variable block size and overlap

In a basic embodiment, the motion block size and overlap are set once for the entire sequence. A number of predefined block/overlap setting are defined for various video standards including; standard definition interlaced, standard definition progressive, 720 line progressive, CIF and 1080 line interlaced.

In some cases, more flexibility and more efficient coding might be achieved by allowing the block size/overlap to vary on a frame by frame basis. The overhead need only be one bit per frame if that bit were used to signal a change. Additional data need only be transmitted when the block size/overlap changed. Note that block size variation, within a frame, is allowed for, to some extent, by the macroblock structure.

Variable Motion Vector Precision.

A simple embodiment may perform motion estimation/compensation to a set precision, for example1/8th pixels/frame period precision. Motion vector precision can dramatically affect compression efficiency. If it is too low the prediction residuals are high, requiring more bits to code them. If vector precision is too high many bits may be required to code them. The optimum vector precision depends on the picture content, image format (e.g. streamed video standard definition or HDTV), the amount of motion, the type of motion estimator etc.

In a development, a scaling factor may be transmitted for each frame and used to scale the motion vectors. The transmitted motion vector would be multiplied by the scaling factor prior to being used for prediction. This would effectively vary the motion vector precision. If the motion vectors were transmitted with integer (pixels/frame) values then a scaling factor of 2^{-3} would provide the same vector precision as currently used in a basic embodiment (1/8). If the scaling factor is limited to powers of 2 (i.e. it simply indicates the position of the binary point), then a vector precision from 2^{-5} to 2^{2} (or other suitable range) could be accommodated by a three bit code.
The implementation complexity is low for this enhancement and the data overhead is negligible. At the same time it may significantly improve compression efficiency for some applications and image formats. In this way the number of applications that can be addressed may be increased.

Motion Vector Tracing

This is a proposal to simplify the design and improve the computational efficiency of the encoder.

Motion vector tracing is a method of tracking the motion through frames or fields. The motion vector of a pixel in the first frame points to a pixel in the second frame whose motion vector in turn points to a related pixel in the third frame. In this way one can move from motion vectors measured across a single frame to motion vectors measured across two or more frames. In general, a frame has different motion vectors pointing backwards and forwards. Given these two sets of motion vector the motion between arbitrary fields can be determined.

To make vector tracing work well a vector refinement stage is required following the basic vector tracing stage. Motion vectors can only, conveniently, be traced using the integer part of the motion vector. Otherwise the projection along a motion vector will not land on a pixel in the following (or preceding) frame. There are also inevitable inaccuracies in the vectors, both from limitations in the measurement process and from any quantisation applied when they are stored. Therefore during vector tracing small errors can accumulate. These can be corrected by performing a final image matching stage after vector tracing. Thus vector tracing provides an approximate estimate of the motion vector and the final matching stage refines this to sub-pixel accuracy. We propose two techniques which might be used to limit error build up in the vector tracing process. Firstly, for estimating between distant frames, refinement stages can be introduced after a fixed number of "hops". Secondly, a simple but effective method is to use error feedback in the quantisation of motion vectors to the integer values needed for vector tracing. Using error feedback should prevent a systematic build up of error during vector tracing and may reduce errors by essentially averaging multiple measurements. For example, without error feedback a constant vector of, for example, 1.3 pixels per frame would be consistently rounded down to 1 pixel per field period resulting in a systematic under estimate of the velocity of 0.3 pixels per field period. However, if you stored the vectors to the original accuracy, the problem wouldn't be so bad: you would just end up with a drift of 0.3 pixels per frame in the location that you were looking, but the accumulated displacement would still be correct.

Vector tracing may simplify the structure of the encoder. The proposal (above) for multiple reference frames makes the codec very flexible and thereby enables improvements in coding efficiency. However this flexibility itself requires a greater flexibility from the motion estimator. Vector tracing allows the computationally intensive part of motion estimation to be partitioned from the rest of the design. The interface to the motion estimator preferably allows access to the sequence of frame period motion vectors. These can then be "rendered", by a relatively simple vector tracing process to produce the appropriate motion vectors between the desired frames. Decoupling the motion estimation from the rest of the encoder algorithm is an advantageous feature; it makes it easier to substitute motion estimators. Alternative motion estimators might yield improved performance or advantages in applications for which the quality/computational complexity trade off is different.

As with any partitioning of system functionality there is inevitably some loss of the ability to optimise the system as a whole. This might apply to the process of rate distortion optimisation, which is discussed below. However it is a trade off between theoretical optimal efficiency and practicality. Good engineering practice is always to partition systems as finely as possible, unless there are pressing reasons not to do so. In this case, if vector tracing does simplify the encoder structure, it is preferable.

The second benefit of vector tracing is improved computational efficiency. With full search block matching the computational complexity increases as the square of the maximum displacement. Searching over a 4 frame period thus requires 16 times the processing power of searching over a 1 frame period. This is a reason hierarchical methods are used for motion estimation, including in the disclosed implementation of the codec. However, hierarchical motion estimation, whilst significantly speeding up motion estimation, may have drawbacks. It might be argued that the solution to searching for a large displacement is simply to increase the number of levels of hierarchy. This has several drawbacks. The lower resolution tiers of the hierarchy have suffered repeated filtering and subsampling and are likely to be degraded. Low resolutions images cannot "see" small objects, which are likely to be neglected. The risk of finding local minima increases with the number of hierarchical levels. All these factors degrade the quality of the motion estimation. In many cases the best solution is to use both a reasonable number of levels for hierarchical motion estimation plus vector tracing. There are, however, some disadvantages to vector tracing, in that one may track background back if an object has passed over it, and may not find a good match that does not happen to follow the tracing-based motion model.

It might be argued that partitioning motion estimation as a separate subsystem precludes the use of rate distortion optimisation. However, rate distortion optimisation can be included, albeit approximately, in the motion estimation process. That is motion estimation can be performed so that it jointly minimises the matching error metric and the entropy of the derived vector field. This is a good way of performing motion estimation in general. Although this is only an approximation to rate distortion optimisation so too are other implementations (and this is not a fundamental flaw), because in practice it is difficult to perform fully rigorous rate distortion optimisation. Rate distortion, for both motion estimation and macro-block splitting, can be included in the refinement stage following vector tracing. In this way the output of vector tracing is effectively used as a sophisticated way to find a candidate vector for the final stage of motion estimation.

In summary the proposal of this section is to apply vector tracing, optionally with vector quantisation using error feedback, for motion estimation. The objective is both to simplify the structure of the coder and by reducing computational complexity, to speed up the encoding process. This feature is optional and does not necessarily affect the decoder. However this sort of approach may be desirable to maximise advantage of the potential of long-term memory frames.

DC coefficient coding

If good predictions are not available, then a block is deemed to be intra coded, and no subtraction is performed for that block in the motion compensation process. Intra blocks have quite different characteristics from other areas of the Inter frame. In particular, their mean will be non-zero, whereas the mean value in other areas will be close to zero. Although the block overlaps will soften the edges of these areas, they will still be coded relatively inefficiently because of the transition in DC value. To deal with this, the codec subtracts the local DC from these areas and codes it separately.

The DC value is computed in the coder from the un-compensated original frame, although the coder is of course free to compute the value in any way. It is subtracted from the Inter frame using the same raised-cosine windowing: intra blocks can therefore be considered as not being intra at all, but as predicted by weighted ‘constant blocks’. Since PREDMODE can apply to blocks, sub-MBs or MBs themselves, depending on MB parameters, the same applies to DC coefficients. Hence DC removal is applied for variable-sized blocks and DC values are coded according to the MB types imposed by the MB parameters.

DC coding operates on the principle established herein of prediction followed by entropy coding. However, both prediction and entropy coding are simpler in this case.

As for MV data encoding, the DC coefficients are scanned by MB in raster order and in raster order within each MB, depending on the MB data and prediction modes. DC coefficients are predicted by some value dependent only on the previous DC value to be encoded. In particular, there is no attempt as yet at neighbourhood prediction. Instead, the prediction for $DC(n)$, the nth DC coefficient so far coded is given by:

$R(n)=DC(n)-d{\cdot}DC(n-1)$ for some value d.

If the block coordinates of $DC(n)$ are designated $(x(n),y(n))$ then the factor d is given by:

$$d=1-\frac{Max(|x(n)-x(n-1)|+|y(n)-y(n-1)|,8)}{8}$$

The purpose of this arrangement is that DC blocks are generally very sparse. There is no point in having a very complicated prediction structure from neighbours that don't, for the most part, have coded DC values anyway. The value of a prediction also recedes with distance. If the last DC block coded was a long way away then using it as a prediction will increase bit rate, not decrease it.

Entropy coding uses the expGolomb VLC for the magnitude bits, which is as shown in Fig. 23.

The rule is that there is a length prefix of N zeroes followed by a 1. Then there is an N-bit binary number b. If $N>0$, the number represented is $2^{N-1}+b$. Sign bits are uncoded.

The method of DC coefficient coding may be empirically adjusted to improve its performance. However, the basic idea is effective and should give some gains on the most difficult sequences, where motion estimation fails.

Prediction of Context Statistics from Previous Frames

The embodiment uses adaptive binary arithmetic coding for entropy coding. The efficiency of arithmetic coding is determined by the accuracy of the "contexts" used. These are the probabilities of the symbol being a one or zero (remember we're talking about *binary* arithmetic coding). At present the probabilities of 0 or 1 are both initially set to ½. The statistics are then updated by counting the numbers of 1's and zeros that are coded. As the count increases the influence of a single sample on the statistics becomes diluted. Periodically the counts are "refreshed" by halving the counts of ones and zeros. This keeps the counts in a reasonable numeric range. This scheme is advantageous in that the statistics rapidly converge to an appropriate value. The down side is that symbols contribute unevenly to the statistics.

A coding gain might be achieved by providing initial statistics for the contexts in adaptive arithmetic coding. Restarting adaptive binary arithmetic coding, with a context probability of ½, ½, has a start up cost of, perhaps, a byte or two per context per restart. This may be appreciated by considering a symbol with a low (or high probability). The first symbol assumes a probability of ½ and so has a coding overhead of about ½ bit compared to coding with correct statistics. The second symbol is coded with a probability 1/3, and so has a coding overhead of about 1/3 bit. The cumulative overhead is the sum of the harmonic series 1/2, 1/3, 1/4, 1/5... In fact this series does not converge but is approximately proportional to log(n). For reasonable numbers of symbols this would give an overhead of one or two bytes per context per restart.. A practical codec may have about 16 contexts for the residues of each subband, although not all these contexts are used for every subband. There may also be forty or so more contexts used for coding the motion vectors. In total two hundred and fifty separate contexts may be used in a typical practical implementation. If initialising each context has a cost of 1 byte then for a 12.5Hz sequence as much as 25kbit/s may be required to initialise all the contexts. In coding HDTV this may be negligible but at the low bit rates used for streaming video it may be worthwhile to try to reduce this overhead.

One way to provide initial values for context statistics would be to use the corresponding probabilities estimated in a pre-decoded frame. Preferably, one takes the probabilities from a frame of the same type. The probabilities for a layer 0 (I frame) are likely to be significantly different from those of a layer 1 or layer 2 frame. Similarly the statistics of a frame predicted from two reference frames are likely to be different from those of a frame predicted from a single reference frame, largely due to the likely number of intra blocks present. In practice this normally means 5 frame types (layer 0, 1 & 2 frames predicted from 1 or 2 reference frames) for the purposes of context initialisation. The statistics will also vary by subband, and preferably one should take the statistics from the corresponding subband in the corresponding frame type.

It is also necessary to consider from which previous frame to take the probabilities. Normally one cannot simply take them from the previous frame because frames will often be encoded out of presentation order. Furthermore frames are often predicted from a reference frame of a different type. We propose that the encoder maintains a list of context probabilities for each type of frame. The list would have approximately twelve hundred entries (250 contexts times 5 frame types). The list can be updated with the context probabilities whenever a subband has been fully coded within a frame. That is, the probability used for initialisation should be the most recently used probability of the same context and frame type. These statistics may not be the "best" initialisers that could be found. However only an approximate initialisation is needed to reduce the bit rate and the proposed scheme is relatively easy to understand and implement.

A slight variation of this scheme, which is potentially more effective, is to save the context probabilities after the first refresh when coding each subband in each frame type. In this way the saved probabilities relate to the beginning of the encoded data, and so will be more appropriate for initialising subsequent frames than those accrued by the end of the encoding process.

Another potential alternative is of explicitly transmitting initial context probabilities. However in many cases, this is likely to use as much bit rate as it saves.

Whilst using context statistics from preceding frames may improve compression it increases the interdependence between coded frames. This may exacerbate the propagation of bit errors. As with adopting a more general prediction structure, it would also be necessary to flag access points into the sequence to avoid having to start decoding from the beginning of the sequence. This flag would imply using a fixed initialisation for the context statistics for the first of each type of subsequent frame. More generally, it may be advantageous to transmit a flag for each frame that indicates whether its context statistics should be initialised using corresponding contexts from a previous frame. This could allow for greater error resilience. The idea could be extended to flag each individual context, but this adds complexity and in many cases is unlikely to yield significant improvements in coding efficiency.

It is a preferred feature to include a flag in the bitstream to indicate that context statistics should be initialised with the values from a previous frame rather than the default (½, ½).

Updating Context Statistics

In one embodiment of the codec context statistics may be updated for every symbol. They are refreshed, by halving the counts, after a fixed number of wavelet residuals have been coded. The symbol count is the total for all contexts rather than being specific to individual contexts. This scheme ensures that context adaptation takes place over a defined portion of the image area.

There are some potential disadvantages to this otherwise simple but effective scheme. The contribution of each symbol to the statistics varies depending on how long it was since the last refresh. Context statistics are re-calculated every symbol with the associated computational overhead. Rarely used statistics are refreshed rather unpredictably. They are not updated if the number of symbols encoded is too low. However this means they are not refreshed at the same time as the other contexts. This unpredictability requires additional tests and branches in a software implementation, which are undesirable for processing efficiency.

Enhancements are possible that would improve the coding efficiency and computational complexity and affect the decoding method.

The uneven contribution symbols to the statistics can be addressed by simple first order recursive filtering. The computational complexity can be low if a factor α is taken of the new probability (1 or zero) and a factor $(1-\alpha)$ of the previous average probability. If the factor α is 2^{-n} then the multiplication is simply a bit shift. However, experiments have shown that recursive filtering often gives worse results than the above simple scheme. This may be because the statistics take a long time to reach an appropriate value - for a value of α of 2^{-n} then the statistics will take on the order of 2^{+n} samples to reach an appropriate value. For this period the arithmetic coding will not be providing efficient entropy coding. However, if the context statistics are initialised to roughly the correct value then recursive filtering could be applied without degrading the efficiency of arithmetic coding. So recursive filtering could be applied in conjunction with initialising context statistics.

The advantage of the above simpler system is fast adaptation, but it may be possible to combine this with recursive filtering. In the basic version, the contexts are initialised to one count for symbol "0" and 1 count for symbol "1". The statistics initially adapt quickly, but this later slows down as the symbol count increases (hence the need for refreshing). A potential drawback of initialisation and recursive filtering, as discussed above, is that even though the contexts are initialised they may be slow to adapt to changing statistics. This problem could be mitigated by adaptive recursive filtering. The feedback coefficient (α) could vary with the symbol count. In one implementation, the context can be initialised to approximately the right value. The symbol count might be reset for each frame (and for each context). When the symbol count is low the feedback factor ($1-\alpha$) should ideally be low allowing fast adaptation. As the symbol count increases the feedback factor should ideally also increase, slowing adaptation. However, the feedback factor should ideally not be allowed to increase beyond a preset limit, ensuring that adaptation never becomes too slow. One way control of adaptation rate with symbol count can be achieved is via a lookup table, which, for an input of the symbol count, provides the appropriate feedback coefficient. If the output of the lookup table is always a value of the form 2^{-n} (n positive integer) then this scheme would be computationally efficient (as described above).

The issue of computational efficiency can be addressed by recalculating context statistics less frequently. The symbol counts can be accumulated for a fixed period but only applied at the end of that accumulation period, thus avoiding symbol rate computation. The accumulation period can be related to the number of values coded. This preserves the link with picture regions embodied in the current refresh strategy. It also minimises computation since the individual contexts do not need to keep track of when the next refresh is due. At the end of the accumulation period the statistics can be refreshed as at present. However the refresh factor need not be ½ as at present but, instead, $(1-\alpha)=(1-2^{-n})$ as described above for recursive filtering. A larger refresh factor allows more frequent updates, combining the advantages of the current scheme with the recursive filtering strategy.

Alternative Contexts for Intra Coded Blocks

Typically not all of a frame in an image sequence can be effectively motion compensated. Areas which cannot be motion compensated may appear at the edge of a picture, in regions of revealed background (although prediction from two reference frames can reduce the area which cannot be motion compensated) or in areas of complex motion. Blocks in uncompensated areas use intra frame coding (i.e. essentially the wavelet transform of the original signal, less its DC value, rather than the prediction residual).

Residuals for intra-coded areas of the picture have different statistics to motion compensated parts of the picture and so in these parts of the picture are not effectively entropy coded because the context statistics do not apply. One possibility is to increase the context adaptation speed, but then the MC predicted areas would have erroneous statistics until they had re-adapted. A basic embodiment typically suffers efficiency losses on the small intra-coded areas, but these are in many cases too small to affect significantly adversely the context statistics for motion compensated regions.

An improvement is to provide separate statistics for predicted and intra coded regions of the picture. In a typical embodiment, the intra regions are typically too small for the context statistics to adapt effectively without initialisation. However initial context statistics can be estimated from previous frames as described above.

The decoder knows the location of all I blocks and so can switch context for these regions of the picture. No additional data need be transmitted to implement this feature. In one implementation, the zero tree map, which is already used to select a residual context, can be extended to indicate whether the block was predicted or intra as well as the parent coefficient magnitude.

A downside of this proposal is increased complexity. Relating intra blocks in the pictures to corresponding regions in subbands is involved. However, in the context of alternative compression systems, for example using lapped orthogonal transforms or MDCTs (modified DCTs), this proposal might be useful.

Version Number

Preferably, a version number is included in the coded bit stream that specifies the version of the codec used to encode the stream. This would (implicitly) specify which tools were used in coding the bit stream. The version number can be considered to equate approximately to “Profiles” in MPEG codecs. However, an important difference is that higher versions should ideally be a strict superset of lower versions to ensure backward compatibility. This would preclude, for example, the use of overlapping block transforms instead of wavelets. In a preferred implementation, a first version number corresponds to a first set of coding possibilities and a second, higher version number corresponds to a second set of coding possibilities, including the first set. A coder may be arranged to code and a receiver arranged to decode accordingly selecting or inferring coding possibilities based on the version number.

Numerical Precision & Drift

If the locally decoded pictures within the decoder do not precisely match those generated by the decoder the predicted pictures at the decoder will start to drift away from those generated by the encoder. This might be caused by slightly different numerical precision within encoder and decoder. In MPEG 2 drift can occur due to slight mismatches between the encoder’s and the decoder’s implementation of the IDCT. In MPEG 2 drift is, ultimately, controlled by specifying that a picture must not be predicted more than 131 times without being refreshed by the transmission of an I frame. Preferably in a practical codec drift is controlled by specifying a maximum number of predictive operations or otherwise making a similar stipulation and/or by specifying the (minimum) precision of arithmetic operations where applicable.

Further developments

The following sections briefly sketch some further ideas in video coding that can be used to develop the above wavelet coding architecture.

Non-linear wavelets

Wavelet transforms can be understood in the context of a device called a lifting scheme. In the lifting scheme, a discrete signal is decomposed into two parts in three stages, shown in Fig. 24. A *splitting* stage does an initial reversible split, which could (for example) just be a split into alternate samples. A *predict* stage then uses one (the lower in the figure) of the resulting streams to predict the other, and this is then replaced by the difference between the sample and its predictor. Using this new sequence the other stream can then be modified by an *update* function. The result, provided that the original splitting process is reversible by some merge function, is itself clearly reversible by performing these operations in reverse order with addition replaced by subtraction.

If the split, predict and update processes are linear, then the result is a perfect-reconstruction filter bank, and hence wavelets. In fact, any orthogonal or biorthogonal wavelet transform can be constructed by iterating lifting schemes, optionally with additional gain multipliers and reversing the order of predict and update.

Example. Suppose the split function simply splits the signal x_n into even and odd samples, x_{2n} and x_{2n+1}. Suppose also that the predict operator is the identity $P(z_n)=z_n$, and the update operator simply divides by 2: $U(z_n)=z_n/2$. Then the resulting transform is (up to a gain factor in each stream) equal to the Haar wavelet filter bank (1/2,1/2) and (1/2,-1/2).

Very complicated transforms can be built up from simple stages very easily, since each Split function can be iteratively composed of a whole lifting stage. The update and predict operators need not be linear. If the initial split is a linear wavelet splitting, then the predict and update functions can be used to correct for the less welcome characteristics of the wavelet filters. This opens the way to non-linear wavelets.

On natural images, wavelet filters must perform a compromise. For relatively smooth areas of the picture, a large number of vanishing moments are desirable so that the signal can be represented compactly as being approximately polynomial. This requires a long filter, however. At edges within the image, long filters are a disadvantage and cause ringing when combined with quantisation. So it makes sense to vary the length of the filter according to the signal characteristics so that both smooth areas and edges can be represented compactly. Possible advantageous implementations include:-

1) A splitting function derived from a short-length wavelet filter bank. So far experiments have been confined to (1/2,1/2), (1/2,-1/2) although slightly longer filters may improve iterability.

2) An edge detector for switching prediction modes, operating on the low-pass filtered and subsampled signal.

3) A predict operator based on high-order polynomial interpolation away from the corners of detected edges, leaving values near the corners (where ringing occurs) unchanged.

There is no update operator in this scheme.

The effect is that information about the high-pass filtered subband is gleaned from the low-pass filtered subband by investigating edges, which are compact artefacts spatially but which spread energy over a wide range of frequencies. This is then used to provide additional prediction of the high-pass band to reduce the number of coefficients that need coding.

In this case the predict operator is only non-linear by virtue of being switchable. Median or stack filters might also prove useful.

Coding interlaced pictures using wavelets

Wavelets are generally used in video coding to transform the whole picture: although it is possible to use them as local block transforms, their coding efficiency tends to be reduced. However, in interlaced video, this appears to limit the application of the wavelet transform to coding either whole frames or whole fields. As a first attempt at interlace video coding, coding frames or fields might be quite effective. However, interlace artefacts (vertical-temporal aliasing) arise through motion, and often local motion at that. We have therefore considered a motion-compensated wavelet decomposition.

The idea is that motion compensation is applied to the first stage of wavelet decomposition only, and then only in the vertical direction. MC is used to shift lines, or parts of lines, from the two fields to compensate for the motion of objects so that vertical filtering can then be applied. This process is illustrated in Figure 25.

This principle, although particularly helpful for interlace pictures may be applied to pictures which have been partitioned in other ways than conventional interlace splitting. The portions may correspond to different spatial subsets (e.g. alternate horizontal or vertical lines, chequerboard patterns etc) and the different portions may correspond to different times or the same time. Where the portions correspond to different times, motion vectors may be estimated but in other cases a simple displacement vector between matching portions which does not correspond to motion but simply to areas of correlation may be estimated. It will be appreciated that there are numerous possibilities for employing the technique; the embodiment will be described in the advantageous context of application to interlace however.

We note that it is known to perform motion compensated filtering using perfect reconstruction filter banks along the temporal axis [S.-J. Choi and J. Woods, 'Motion-compensated 3-D subband coding of video', IEEE Trans. Image Processing, Vol 8 No 2, Feb 1999]. However, this is complex whereas in the present embodiment we provide a motion-compensated *vertical* filter within the same frame; this feature may be provided independently.

The filtering process works as follows. Field 1 is used as a temporal reference for motion estimation of Field 2, but the only motion vectors considered are horizontal vectors shifting lines (or parts of lines) in Field 1 relative to the corresponding line in Field 2. Suppose that we consider the lines of the frame numbered in consecutive order from 0 with Field 1 and Field 2 lines alternating; suppose also for convenience that the top line is from Field 1, and so all Field 1 lines are even and all Field 2 lines odd (other orderings and numberings will be very similar in effect). Each sample $x_{m,2n+1}$ on line $2n+1$ of Field 2 is therefore supplied with a predictor

$x_{m+v(m,2n+1),2n}$

from Field 1. The decomposition high-pass output is:

$(x_{m,2n+1}- x_{m+v(m,2n+1),2n})/2$

The low-pass filter is slightly more complicated, since each pixel in Field 1 may predict none (if the pixel is concealed), one or more than one of the pixels in Field 2. The rule is that:

i) if $x_{m,2n}$ predicts no pixels in Field 2 then the low-pass output is $x_{m,2n}$;

ii) if $x_{m,2n}$ predicts one or more pixels then the output is one half of $x_{m,2n}$ and half the average (possibly weighted) of all of those predicted pixels. So that in the case that $x_{m,2n}$ predicts just one pixel $x_{p,2n+1}$ then the output is:

$(x_{p,2n+1}+ x_{m,2n})/2$

If it predicted two pixels, $x_{p,2n+1}$ and $x_{q,2n+1}$, then the output is:

$((x_{p,2n+1}+ x_{q,2n+1})/2+ x_{m,2n})/2$

(Weighted averages might be used in the case that overlapped blocks were used for motion estimation and compensation.)

To recover the original values, one reverses this process. If each point in Field 1 predicts at most one point in Field 2, this is simple, but it is a little more complex if it predicts more than one. Suppose that both $x_{p,2n+1}$ and $x_{q,2n+1}$ are predicted by $x_{m,2n}$. Then the high pass outputs for $(p,2n+1)$ and $(q,2n+1)$ are:

$(x_{p,2n+1}- x_{m,2n})/2$

and

$(x_{q,2n+1}- x_{m,2n})/2$

respectively.

The low-pass output for $(m,2n)$ is $((x_{p,2n+1}+ x_{q,2n+1})/2+ x_{m,2n})/2$, and adding this to the *average* of the two high-pass outputs yields:

$((x_{p,2n+1}+ x_{q,2n+1})/2+ x_{m,2n})/2+((x_{p,2n+1}- x_{m,2n})/2+(x_{q,2n+1}- x_{m,2n}'/2)/2$

which is $(x_{p,2n+1}+ x_{q,2n+1})/2$. From this $x_{m,2n}$ and so $x_{p,2n+1}$ and $x_{q,2n+1}$ can be determined.

The case of more than one pixel in Field 2 being predicted by a pixel in Field 1 can be avoided by having only a single motion vector per line, and this would be the simplest approach (which is surprisingly effective) for an initial implementation.

After this initial vertical decomposition, the rest of the wavelet decomposition can be performed without any further motion compensation.

Developments

The basic principle can be modified by in several ways. For a start, longer filters could be used. The description given above is for motion compensated (1/2,1/2) and (1/2,-1/2) filters. However any pair of half-band perfect reconstruction filters could be used. Suppose that $g(k)$ and $h(k)$ were such a pair, with $g(k)$ the low-pass and $h(k)$ the high-pass filter. The result of vertically filtering a picture by $h(k)$ using motion compensation is given by using the real sample for each sample in Field 2 and using a motion-compensated sample for samples in Field 1. This means that the output at line $2n+1$ is:

$$\begin{aligned}\tilde{x}_{r,2n+1} = (h *_{mc} x)_{r,2n+1} &= \sum_k h(2k) \cdot x_{r,2n+1-2k} + \sum_k h(2k+1) \cdot x_{r+v(r,2n-2k+1),((2n+1)-(2k+1))} \\ &= \sum_k h(2k) \cdot x_{r,2n+1-2k} + \sum_k h(2k+1) \cdot x_{r+v(r,2n-2k+1),2n-2k}\end{aligned} \qquad (*)$$

The low-pass filter could be constructed in a similar manner to that described above:

$$\tilde{x}_{r,2n} = (g *_{mc} x)_{r,2n} = \sum_k g(2k) \cdot x_{r,2n-2k} + \sum_k g(2k+1) \cdot \bar{x}_{r,(2n-(2k+1))} \qquad (**)$$

where $\bar{x}_{p,2q+1}$ denotes the average (possibly weighted) of all those samples $x_{w,2q+1}$ in Field 2 whose motion-compensated predictor in Field 1 is $x_{p,2q}$.

Aside from longer filters, vertical motion compensation could also be incorporated, and the formulae given can be generalised to this case: for example, (*) would be modified to be:

$$\tilde{x}_{r,2n+1} = (h *_{mc} x)_{r,2n+1} = \sum_k h(2k) \cdot x_{r,2n+1-2k} + \sum_k h(2k+1) \cdot x_{r+v1(r,2n-2k+1),2n-2k+v2(r,2n-2k+1)}$$

The idea is not limited to the particular description given: other methods of performing motion-compensated perfect reconstruction filtering could be used. Also, the roles of Field 1 and Field 2 can be interchanged in the previous description, as can the roles of even and odd lines. Although conventional interlaced video invariably has two fields, the techniques can be applied to more than two fields per frame were this considered desirable.

Interlaced Video

Even though many images will be progressively scanned, coding of interlaced video is important as there are vast archives of video content. There are various options for coding interlaced video. The simplest option is simply to code interlaced frames as if they were progressive. This is a relatively inefficient coding strategy, which was used in MPEG 1. One embodiment has the option of coding the entire sequence as interlaced. This means that fields are treated as frames except that odd fields are predicted from odd fields and even fields from even fields. The exception is for I frames in which the second field is predicted from the first. Experimentally this seems to offer little coding gain versus the simplest approach, except for coding I frames, for which there is a benefit.

There are several possible alternative modes for coding interlaced.

One possible interlaced mode is to add an additional interlaced prediction mode. Each block may still have a single motion vector, measured in units of pixels per frame the same as for progressive prediction. Fields within the predicted frame can be predicted from the fields within the reference frame(s). One option is to predict fields only from fields of the same parity. This is subtly different from the current progressive prediction mode because odd field lines will only be predicted from odd field lines and even field lines from even field lines. There will be no difference in the prediction for integer pixel motion, but sub pixel motion will be predicted differently due to interpolating field, rather than frame, lines.

Fields are not always best predicted from fields of the same parity. With vertical motion of an odd number of frame or picture lines per field, fields would be better predicted from fields of the opposite parity (a field could be predicted from either odd or even field or, more generally, a weighted sum of predictions from the two field parities but there is a shorter delay between fields of opposite parity and this make it a better choice for prediction). So the best field parity with which to predict a field depends on the vertical motion speed. More generally a field would be best predicted by a weighted sum of predictions from fields of both parities. The weighting would vary with vertical velocity. For vertical speeds of an even number of lines per field the weighting should be 100% contribution from the field of the same parity and 0% from the field of the opposite parity. Conversely for vertical speeds of an odd number of lines per field the weighting should be 100% contribution from the field of the opposite parity and 0% from the field of the same parity. For vertical speed of a half integer number of lines per field symmetry indicates the weighting should be 50% from each field.

A good prediction of an interlaced field is a weighted sum of the predictions from both fields of a frame. Let the motion compensated prediction of an even field from the even field of a frame be E_e and the predication of the even field from the odd field of the frame by E_o. Then the overall prediction of the even field, for a velocity v measured in lines per field, would be;

$$P_e = \alpha(p)E_e + (1-\alpha(p))E_o \quad ; p = v \bmod 2$$

where p is the least significant integer bit plus the fractional part of the velocity. The weighting function α should be 1 for p=0, 0 for p=1 and 0.5 for p=1/2 or 3/2. These constraints suggest that a suitable definition of α might be the raised cosine function;

$$\alpha(p) = \frac{1}{2}(1+\cos(\pi p))$$

although other functions might also be suitable. Similarly the prediction of an odd field would be given, with analogous notation, by;

$$P_o = (1-\alpha(p))O_e + \alpha(p)O_o \quad ; p = v \bmod 2$$

The motion vector is preferably scaled appropriately to find the displacement to use for the motion compensated prediction. However it is the velocity, not displacement, that is used to determine the required weighting factor. For bi-directional prediction a total of four separate field predictions would be combined to produce the overall prediction.

Introducing an additional interlaced prediction mode introduces minimal changes in the overall structure of the decoder. Frames are always treated on a frame basis, with blocks of the same size, irrespective of whether they are progressive or interlaced. This makes it simple to change between interlaced and progressive prediction. There would only be a single motion vector per block and this would be scaled the same (i.e. in pixels per frame) for both progressive and interlaced modes. This interpolation mode could be applied either on a frame by frame basis or on a block by block basis. However, block-by-block adaptation is more complex and may yield little coding improvement. So frame-by-frame adaptation may be preferable and this may be provided as an independent feature.

A second possible interlaced mode would be simply to treat interlaced frames like a pair of sequential, half height, frames. This would differ from the basic codec in two respects. Each frame could be independently flagged as interlaced or progressive, so that the whole sequence did not have to be coded one way or the other. Secondly fields could be predicted from other fields of either parity. Although this is more complex that the first option this is still relatively simple to implement and has advantages.

A third possibility is to predict the first field of a pair from other fields using motion compensation. Then predict the second field from the first field, using either motion compensation or simply spatial interpolation. This amounts to deinterlacing the first field to predict the second. This is also relatively simple but in some practical cases may be more complex to implement than the first option.

Other video codecs adapt to interlace on a block by block basis. This can be very complex to implement. In practice it may yield small coding gains because of the overhead needed to signal whether blocks are interlaced or progressive. Given that frame cannot normally be both interlaced and progressive it is proposed that many implementations would reject such schemes on the grounds of excessive complexity.

A preferred arrangement is to use the first interlaced option, i.e. a specific interlaced prediction mode as specified above.

<u>Coding of inter-frame residuals</u>

Most frames in any video coding scheme are likely to be inter frames, and most of the bit rate is due to them also, given a reasonably long GOP. Their efficient coding is therefore very important, and in the current scheme there is probably more scope to reduce their bitrate than that of other components of the bitstream.

Inter frames can be very variable in character. When prediction is good, the predicted data consists just of low-level noise. With 1/4 or 1/8 pel accuracy, this noise is at a very low level indeed, and will almost certainly be quantised away. Areas that are predicted poorly can be of two basic types: revealed/concealed areas and non-translational motion areas. It is likely that a good motion estimator will code the former areas as intra, so they will be picture-like in character. The latter areas may sometimes be matched with similar areas and sometimes not. If predicted the residual can look quite random, although often very 'edgy' and certainly not white-noise-like.

The wavelet transform still does a good job of decorrelating these poorly predicted areas. However, the appropriate quantisation level in different parts of the picture may be quite different. In a basic but effective implementation, the coder defines a single quantiser for each subband. More quantisers could be defined by splitting each subband into blocks and this would be an advantageous next step.

A more complicated but potentially adaptive idea uses the parent-child relationship that wavelets can introduce. As can be seen from the Figure 26 below, areas which are not well predicted tend to produce significant coefficients across all subbands, whereas areas that are well predicted tend to produce low-level, independent noise across all subbands.

The large coefficients tend to be organised in a parent-child arrangement, whereby the pixels corresponding to the same picture area with the same orientation (LH,HL or HH) are related, as illustrated in Figure 27.

Although parent and child pixels are likely to be decorrelated, they are not statistically independent, and it seems that the magnitude of a parent and its children are related. One way of exploiting this is to partition the coefficients in each subband (other than the DC band) according to size. This partition can then be inherited by the child subband via the parent-child relationship, and a different quantiser assigned to each partition. The important point is that if the partitioning process is performed on *quantised* coefficients in the parent subband, then it can be performed by the decoder in an identical fashion. There is then no need to signal the shape of the different partitions of the subbands to the decoder, and the only overhead, therefore, is signalling the additional quantisers.

Implicit classification and quantisation

As described above, the parent-child relationship available in wavelet-transformed data can be used to partition data implicitly for quantisation. When quantising a subband, the data in the parent subband can be classified by whatever means is desired into a number of sets and this classification then inherited by the child subband. Each subset of the band can then be given a separate quantiser. If the parent band has been encoded before the child band then the classification can be performed identically at the decoder. A different quantiser can then be selected by the encoder for each defined subset of the child band, and transmitted to the decoder, with no need to transmit the classification of child subband pixels. There are some important considerations:

1) The classification need not, or not only, be used for defining quantisation classes, but could also be used for defining contexts for entropy coding.

If a given set of coefficients have been quantised differently from another set of coefficients within the same subband, then it is likely that the statistical properties of the two sets will differ and different statistical contexts would be appropriate for entropy coding for the two sets, especially for adaptive arithmetic coding. Even if the same quantiser was selected for all sets (ie the classification was not used for defining quantisation classes) a partition of coefficients in the parent subband (on the basis of coefficient statistics or other means) could be used to define different coefficient classes for entropy coding, and could prove useful.

In so-called zerotree coders [A. Said and W. Pearlman, 'A new, fast, and efficient image codec based on set partitioning in hierarchical trees', IEE Trans Circ. Syst. for Vid. Tech., Vol 6, No3, June 1996], significance data – whether a coefficient is larger than a given threshold – is often coded conditionally on whether the parent coefficient is significant or not, but quantisation is not affected, and general-form partitioning into multiple sets has not been entertained.

2) One useful classification is to classify coefficients in the parent subband into subsets with different variances or other statistical properties.

In particular, one suitable simple classification is into zero and non-zero coefficients, or into zero, small, and large coefficients. Local properties can also be taken into account, for example local masking so that a class of coefficients defined in this way can be quantised more heavily if the results are likely to be masked. Another case where this might be useful would be Region-of-Interest (ROI) coding where coefficients corresponding to more important areas of the picture (such as human faces) might be less heavily quantised.

To facilitate the exploitation of local features, the classification into subsets, although implicit, could also be augmented, refined or amended with additional signalling from the encoder so that a finer classification could be obtained if the implicit one proved insufficient. For example, if coefficients are split into two classes: has_zero_parent and has_non-zero_parent, then the encoder could signal that has_non-zero_parent should be further split into, say, coefficients correponding to central areas of a picture and those corresponding to peripheral areas. Certain coefficients could even be identified and moved from one class to another.

3) The principle is not limited to two dimensional transforms. It can be applied to one-dimensional signals (for example in audio coding): in this case each parent coefficient has two child coefficients. It can also be applied to 3-dimensionals signals, for example a 3D-wavelet transform, motion-compensated or not, of a video signal. In a 3D transform, each coefficient has 8 children in the child subband.

4) Wavelet transforms are not the only transforms for which parent-child relationships can be defined. Other transforms, for example block transforms, Lapped-Orthogonal Transforms, wavelet

packets and even adaptive so-called 'best-basis' transforms can be given a parent-child relationship [Z Xiong, K Ramchandran and MT Orchard, 'Wavelet Packet Image Coding Using Space-Frequency Quantization', IEEE Trans. Image Proc, Vol 7, No. 6, June 1998.], and the method described above can be applied.

Block transforms, such as the LOT or the widely-used DCT, can be given a parent-child structure in a number of ways [T. Tran and T. Nguyen, 'A progressive transmission image coder using linear phase uniform filterbanks as block transforms', IEEE Trans. On Image Proc., Vol 8 No 11, Nov 1999].

To understand how this might be done, it is helpful to think of block transform coefficients as the outputs of critically-sampled perfect-reconstruction filterbanks, just like wavelet coefficients, only with all subbands having the same bandwidth and being subsampled by equal factors. This can be done by conceptually grouping coefficients corresponding to the same area of the picture. So the (i,j)-th coefficient of block (p,q) can instead be thought of as point (p,q) within frequency subband (i,j) – see Figure 31 in which a 4x4 array of 2x2 transform blocks can be considered as a 2x2 array of 4x4 subbands.

Given this rearrangement, Figure 32 shows two possible parent-child relationships, the first a wavelet-like relationship. In this case a coefficient may have children in different bands.

5) The classification can be used for defining quantisers as well as quantisation classes.

An analysis of the statistics of the parent band can allow the quantiser values themselves to be determined implicitly also, avoiding the need to signal them to the decoder.

A representation of the partition and quantisation algorithm can be defined as follows. Given a subband B, let $P(B)$ denote the parent subband to B. If $\mathbf{c}$ represents the coordinates of a coefficient in B, let $p(\mathbf{c})$ in $P(B)$ denote the parent coefficient coordinates; if $v(\mathbf{c})$ denotes the value of the coefficient at $\mathbf{c}$ then let $q(v(\mathbf{c}))$ denote the quantised value.

```
1.      For every subband $B$, if $B$ has a parent band that has already been encoded,

DO

{       1.1     Set $P=P(B)$
        1.2     Define sets $P_1, \ldots, P_k$ such that
                        a) $P_i \cap P_j = \varnothing$ for all $i, j$
                        b) $\bigcup_{i=1}^{k} P_i = P$
                according to a predefined rule based on the quantised coefficients
```

```
$q(v(\mathbf{c}))$ $\mathbf{c} \in P$
1.3     Define sets $B_1, \ldots, B_k$ by the rule $\mathbf{c} \in B_i$ if and only if $p(\mathbf{c}) \in P_i$. For each set $B_i$
DO
{       1.3.1  Pick a quantiser $q_i$ for $B_i$.
1.4.1   For each $\mathbf{c} \in B_i$, quantise $\mathbf{c}$ by $q_i$.
}
}
```

This implicit adaptive quantisation can therefore be used to match the quantisation levels to the shape of the residual data. The approach may have a drawback that subbands cannot be decoded independently and this is often useful, but this is not always essential, and other methods of parallelisation may be employed.

Other coding developments - Multiple Quantisers within a Sub-Band.

In one embodiment of the codec different quantisers (i.e. different bin sizes) are used depending on whether a coefficient's parent is zero or non-zero. This feature is found to yield a 2% reduction in bit rate but perceptual modelling might enhance the gain. A simplified codec may not include this feature.

The performance improvement of different quantisers is modest but so is the complexity, and there is little impact on computational requirements so in many applications the feature is preferred.

It would be possible for different regions of the picture to use different quantisers. This would allow for spatially varying picture statistics. In particular intra blocks, within inter frames, are likely to have significantly different statistics and to require different perceptual weighting. However the quantiser regions would be of different sizes in different sub-bands and would not be aligned with the motion compensation blocks.

Splitting the picture into regions may be beneficial. Each region can have its own quantiser(s) and associated arithmetic coding context(s). Although the regions would not (necessarily) align with motion compensation blocks the coder could determine whether it was beneficial to treat the image as a single region or divide it into smaller regions.

There are implementation advantages to splitting subbands, in addition to potential coding gains. Smaller regions fit more easily into processor caches and so improve decoder speed. One possible implementation uses region based quantisers for all parts of the picture, not just intra blocks. In such an implementation the quantisers could be coded differentially (with entropy coding) requiring only a small data overhead. In fact, with this implementation, one could choose all the quantisers to be the same with a minimal overhead in data rate. In this way the use of multiple quantisers in a subband can be viewed as a pure superset of the coding options in a basic implementation.

Implementations of multiple quantisers schemes do however introduce additional complexity and in many cases the gains may not be significant to justify this.

Thus, it is considered generally advantageous (although of course not necessary) to provide a first quantiser for the case where parent has a zero coefficient and a second quantiser for the case where the parent has a non-zero coefficient.

An optional feature is the use of different quantisers and contexts for different regions of the picture; this feature may be beneficial in some implementations.

Global motion compensation

This is a simple idea to implement and is suitable for pans, tracking shots and especially for camera rotations and zooms. In these cases it makes sense to remove the global motion by globally motion-compensating the reference frame before using it to encode translational motion. In the case of a zoom, the magnification factor can be extracted and used to expand/contract the reference frame to match the current frame; the remaining translational motion will be small and both the motion vectors and the residual will be relatively easy to code.

A more complicated idea was used in an early proposal for H.264. In this case, general warping parameters were defined for the reference frame and the resulting reference used for translation motion compensation. This process can help remove many local sources of prediction error, such as figures walking towards a camera, or objects rotating, which cannot be eliminated in translational motion modelling.

It has been suggested that the global motion compensation could be applied to the reference picture before estimating residual translational motion on block by block basis. An affine transform of position could describe global motion comprising pans rotations and zooms. An early proposal for H264 used a more complex pre-warping of the reference image, but this idea was not included in the final standard. The disadvantage of pre-warping the reference frame is both conceptual and computational complexity.

An alternative to pre-warping the reference frame, of sending global motion parameters in addition to block motion vectors, is proposed. In this scheme parameters are sent describing the global displacement (motion) between reference and predicted frames. In contrast to pre-warping, suggested above, the reference frame remains unmodified. The block motion vectors are transmitted as an additive correction to the global motion. The advantage of this is that the motion of simple scenes (e.g. combinations of pans and zooms) could be coded in very few bits. Furthermore in many scenes much of the motion is of the background. In such scenes the motion of background regions can not only be transmitted with fewer bits but may also be more accurate and less noisy. Small regions of differential motion (e.g. people in the foreground) can also be coded efficiently using differential block motion.

The alternative scheme achieves much the same result as pre-warping without the computational overhead. This scheme has little conceptual complexity. There is no need to think about pre-warping. Global motion parameters may be regarded as simply an efficient way of coding scene motion using a parametric description. All that is needed is to calculate the global motion offset from each block from the position of the block. The potential advantage of pre-warping is that the continuous change of a motion vector over a block brought about by a zoom or rotation can be corrected for. However, if a small block size is used, this benefit will be minimal and the computation overhead is high.

With an affine transform, a naïve implementation would calculate the displacement for each pixel or block of an image using 1 vector multiplication and addition per pixel or block, according to the equations below. This equates to 2 scalar multiply accumulate operations per component, that is a total of four scalar multiply accumulate operations per calculated motion vector.

Consider the displacement equation:

$$\mathbf{d} = \mathbf{a} \cdot \mathbf{s} + \mathbf{p}$$

where d is displacement, s is the position, a is the transform matrix and p is the pan. In component terms we have:

$$\mathbf{d} = \begin{bmatrix} d_x \\ d_y \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

$$\mathbf{a} = \begin{bmatrix} a_{x1} & a_{y1} \\ a_{x2} & a_{y2} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_x & \mathbf{a}_y \end{bmatrix} \text{ where } \mathbf{a}_x = \begin{bmatrix} a_{x1} \\ a_{x2} \end{bmatrix} \ \& \ \mathbf{a}_y = \begin{bmatrix} a_{y1} \\ a_{y2} \end{bmatrix}$$

By reorganising the equations, we can reduce the number of calculations to generate each motion vector from the affine description from four MAC operations to only two additions per vector (plus a small overhead calculation). If we consider the difference in displacement between neighbouring horizontal blocks is given by;

$$\Delta \mathbf{d} = \mathbf{a} \cdot \Delta \mathbf{s} = \mathbf{a}_x \cdot \Delta x$$

hence;

$$\mathbf{d}_n = \mathbf{d}_{n-1} + \mathbf{a}_x \cdot \Delta x$$

similarly for vertically neighbouring blocks,

$$\mathbf{d}_n = \mathbf{d}_{n-1} + \mathbf{a}_y \cdot \Delta y$$

So, to calculate the displacements the algorithm is first to calculate the displacement of the first block in the image. Then the displacement of the next horizontal and vertical blocks can be found with the equations above and this can be repeated to find the displacements of all blocks in the picture. The incremental displacement can be pre-calculated. Hence calculating the displacement of all the blocks in the picture in this way only requires two additions per block, plus a (negligible) overhead. That is to say, the displacement of a second block may be calculated based on the displacement of a first block and a transformation component, here simply by adding the components.

A potential difficulty with using global motion parameters is the accuracy to which the motion vector of each block is calculated. Using either direct calculation or the iterative scheme above there is the potential for small errors in the calculation of the motion vectors. These errors could accumulate leading to drift in the decoder (see above). To prevent this problem the accuracy to which the displacement, in the above equations, is calculated must be carefully specified.

It may be useful to flag that only the global motion parameters should be used. That is no block motion vectors would be transmitted. This may reduce the bit rate for scenes that comprise only global motion. It might also be used for scenes with only a small proportion of non-global motion. In this latter case the increased bit rate due to incorrect motion compensation for small regions of the image may be more than offset by not having to transmit any block motion vectors. It may also be useful to flag the (common) case of zero global motion. These issues are addressed more fully in the section on "Skipped Frames"

It is preferred to enable the use of global motion parameters. Preferably, the method transmits a flag to signal no block motion vectors and/or transmits a flag to indicate global motion is zero.

Proportional weighting for motion compensation

The bi-directional prediction performed within the coder uses a weighting of (0.5,0.5) for each of the reference frames. One feature introduced into H.264 is the ability to apply different weights to reference frames, to compensate for fading/cross-fading effects. H.264 allows for a linear calculation of weights based on frame distance as well as direct calculation.

The effect of choosing different weights can be quite dramatic when there are cross-fades or major lighting changes. However, the problem is in detecting these occurrences and addressing them properly. Brute-force motion estimation, in particular, is likely to lead to errors even if a cross-fade has been detected.

The wavelet coder uses a hierarchical block-matching algorithm with DC-removal. This means that motion estimation is improves for fades and cross-fades. It also gives an opportunity to detect these effects with very little complexity: since a low-resolution version of both the reference frame(s) and the current frame are available, a simple correlator can be used to determine the correlation between them. If there is a significant imbalance between the correlation factor between the first reference and the current frame, and that between the second reference and the current frame, then the ratio of the correlation factors can be used to weight the references.

For some types of image sequence, e.g. fades, it may be beneficial to predict a frame from a reference frame in which the pixel values have been scaled. In a fade to black, for example, a predicted frame would be dimmer than a preceding reference frame and so the reference should be scaled towards zero.

Fades and other type of sequence would be better predicted by transmitting a scaling factor to be applied to each reference frame. At most two scaling factors would be needed per frame and these would require only a few bytes per frame. Default values of unity (single sided prediction) or 0.5 (bi-directional prediction) could be signalled with a single bit per frame. Potentially significant bit rate reductions might be achieved for some scenes. This assumes the "motion estimator" in the encoder could determine both the appropriate motion vector and the scaling factor. One way to do this would be by increasing the dimension of the search space (i.e. trial and error). Although this would be computationally expensive at the encoder it would, nevertheless, provide benefits, for minimal computational cost, at the decoder. However, more computationally efficient solutions may be possible for the encoder

Typical video compression algorithms give equal weighting to both predictions in a bi-directionally predicted frame. Sometimes the predicted frame is closer to one reference frame than the other. In this circumstance it makes sense to weight one prediction more heavily than the other; the weighting of the two predictions should be α and (1-α). So this is another instance where scaling pixel values in motion compensated predictions may be beneficial.

More generally there is no reason why weighting factors should necessarily be positive or, in the case of two weights, that they should sum to 1. Consider, for example, the case of a fade to black where both reference frames preceded the predicted frame. In this case the sum of the weights would be less than 1 to allow for the overall reduction in brightness.

In one implementation therefore, that coder may, optionally, allow the transmission of a scaling factor for each reference frame used in a prediction. This enables the encoder to allow for both fades and the relative distance between the predicted and multiple reference frames. A flag in the bitstream can enable this feature; otherwise default weightings for the predictions would be used.

The concept of weights could, potentially, be extended so that different weights were provided on a macro block or block-by-block basis. However this is complex to implement and unlikely to yield large coding gains.

Preferred features therefore include (1) providing a weighting factor for the prediction from each reference frame; (2) including a flag to signify the use of default or variable prediction weightings.

Frame interpolation

A frame interpolator can be used to increase the decoded picture frame rate. A commercially available tool (from Microsoft Windows Media Player), based on gradient-flow techniques, not on the motion vector field can be computed in real time on a moderate PC for up to CIF resolution, although it possess artefacts when the motion estimation was not adequate, for example a fly-over of a mountain in which the revealed area behind the mountain was not present in the preceding frame and a different shape from the same area in the succeeding frame.

An inventive development, to mitigate the problems of the existing tool, is based on the fact that in most cases of Internet delivery, a higher frame-rate original is available to the encoder. A typical example would be standard definition material which had been converted to CIF at 25 frames/sec progressive, and then encoded at 12.5 fps by dropping frames. The function of the decoder frame interpolator is to recover the original 25Hz frame rate.

If this is the case it would be possible to perform the same frame interpolation in both the encoder and the decoder, and for the encoder to compare the result with the actual frame and send a correction signal. One possible signal would be the difference, but we have appreciated that this might not necessarily be the best because the interpolated frame, although plausible, might be quite different from the actual one – for example through a disparity in actual and predicted global motion. It would be better only to send a correction for areas which can be identified as revealed and to compensate for any global motion disparity in correcting them.

There are a number of possible cases of adjusted interpolation, in which information is sent to aid interpolation, including but not limited to:-

i) Sending a difference between the residual picture and the corresponding interpolated picture (ie the difference between true and interpolated pictures);

ii) Sending a difference between the suitably scaled motion vector field between members of the subset of pictures used in deriving the interpolated picture, and the measured motion vector field between the residual picture and members of the subset of pictures (ie the difference between true and interpolated motion);

iii) Sending motion vectors to adjust one or more of the members of the subset of pictures before using them for interpolation;

iv) Sending a set of guide motion vectors for use in the interpolation procedure to improve accuracy;

v) Sending parameters to otherwise control or improve the accuracy of the interpolation procedure, or to control its complexity.

Skipped Frames

At low bit rates "skipping frames" can, apparently, yield significant coding gains. Usually frame skipping simply means repeating the previous frame, which might be done immediately after a picture cut and/or to reduce the bit rate e.g. to prevent buffer overflow.

If we implement a global motion vector (see above) then several frame skipping modes are possible. Firstly we could use "classic" frame skipping, i.e. repeating the previous frame. This essentially means transmitting no information about the frame. Secondly we could use only the global motion information to predict a frame. Thirdly we could use both global motion parameters plus (differential) block motion vectors. Fourthly and finally we could use all the motion information plus prediction residuals (i.e. normal, non frame skipping, prediction).

Some of the frame skipping modes would be possible without explicitly flagging them. For example prediction residuals for a subband can already be flagged to indicate zero for the whole subband. Hence a complete frame of zero residuals may be coded efficiently in relatively few bits. Similarly a frame of zero block motion vectors could also be coded in relatively few bits.

Explicitly flagging frame skipping modes would, nevertheless, provide a small coding gain. Perhaps more significantly it would clarify the coder decisions, making the system easier to understand. One way to implement frame skipping in a consistent way is to use a specific variable length code to indicate "no more data for this frame". Placed immediately after the start of a frame this would imply "classic" frame skipping. Placed after the global motion parameters it would indicate the second frame skipping option above. Similarly it could be placed after the block motion vectors (third frame skipping option), or not used at all. The "no more data" flag might simply be the code for the start of a new frame. The use of such a variable length code makes assumptions about the structure of bitstream. In particular that there is at least some use of variable length codes, even if only in frame headers. This is discussed more above.

A preferred feature is to provide a flag indicating no more data for the current frame. This would enable multiple frame skipping modes.

Coder

In a preferred implementation, adjustable parameters may include one or more of the following:
-Lagrangian QP for I-frames
-Lagrangian QP for L1-frames
- Lagrangian QP for L2-frames
- Lagrangian parameter for motion estimation.
- Cycles per degree in video standard at assumed viewing distance for perceptual weighting.
- Luminance only coding
- Width of block used for OBMC, specified for luma file
- Height of block used for OBMC, specified for luma file

- Horizontal separation of blocks used for OBMC. Must be < block width for both luma and chroma, taking into account chroma subsampling.
- Vertical separation of blocks used for OBMC. Must be < block height for both luma and chroma, taking into account chroma subsampling.
- Number of L1 frames in GOP
- Separation of L1 frames from nearest I or L1 frame in GOP
- Specifying that L1 frames are forward-predicted only

Decoder

In a preferred implementation, all the relevant encoder adjustable parameters, and any other information necessary for successful decoding of the bitstream, will be conveyed to the decoder by means of suitably encoded header parameters embedded in the bitstream.

The parameters may include one or more of the following:
- Luminance only decoding
- Width of block used for OBMC, specified for luma file
- Height of block used for OBMC, specified for luma file
- Horizontal separation of blocks used for OBMC. Must be < xblen for both luma and chroma, taking into account chroma subsampling.
- Vertical separation of blocks used for OBMC. Must be < yblen for both luma and chroma, taking into account chroma subsampling.
- Width of luma picture
- Height of luma picture

Additional parameters may include:
- Number of frames to be decoded
- the option to specify 444 format chroma sampling
- the option to specify 422 format chroma sampling
- the option to specify 420 format chroma sampling
- the option to specify 411 format chroma sampling
- Number of L1 frames in GOP
- Separation of L1 frames from nearest I or L1 frame in GOP
- Specifying that L1 frames are forward-predicted only

Bit Stream Syntax

The bit stream syntax may be defined as desired. The syntax to a large extent determines the features that can be used in a practical codec system and the considerations here provide guidance to the definition of a practical syntax.

A couple of features of the syntax were considered in conjunction with the other ideas discussed here.

Preferably headers are provided for each frame.

Some systems, notably, H264 make extensive use of arithmetic coding even for "header parameters". Whilst this may yield a small saving in bit rate it makes parsing the bit stream difficult.

The MPEG 2 (elementary video) bitstream uses a system of "start codes". These are a 24bit header (000...01) followed by another byte (which identifies the type of start code). The start codes are guaranteed not to occur elsewhere in the bitstream and so are easy to parse.

A typical implementation of the present codec cannot guarantee unique codes as is done in MPEG because it uses arithmetic coding. However for header parameters a system of variable length codes is easier to use and parse than arithmetic coding.

One possible implementation uses a combination of these two techniques. Arithmetic coding may be used for large blocks of data (e.g. subband coefficients). Variable length codes may be more appropriate for diverse small pieces of data (e.g. header parameters).

Adding pointers at the beginning of arithmetic coded blocks, which point to the end of the block, would make parsing easier, quicker and increase robustness.

One proposal is to include blocks of "header parameters" at appropriate intervals in the bitstream.

Another proposal is to use a combination of arithmetic coding and variable length codes in the bit stream syntax.

It is preferable to provide pointers at the beginning of (some) arithmetic coded blocks, which point to the end of the block, to make navigation easier.

Indexing of coded media

The problems associated with random access to compressed data streams are discussed above in relation to the novel coding scheme but also apply to other coding schemes. For some coding schemes, and particularly for short data streams, it may be possible to provide a complete index table for the data, for example in a global header for the data stream. However, such a header is likely to be impractical for anything other than a short data stream (for which random access capabilities are less useful) and may preclude the use of the data stream for streaming applications. In addition, such a system would require the header at the start of the data stream to be accessed for every access of the data stream.

One embodiment of a data stream produced according to the methods described herein is shown in Fig. 33. The data stream illustrated is a media data stream comprising a plurality of frames of media data, for example video data. The media data may be compressed using the compression techniques described herein but similar considerations apply with MPEG compression or other known compression techniques.

During the encoding, access points 114 may be identified. The access points may comprise points, for example frames, at which a user can start to access data in the data stream without requiring reference to other points in the data stream, which will include layer 0 frames and context refresh points. In another application, access points may comprise I-frames in an MPEG-compressed data stream.

Once the access points have been identified, index data 110 may be placed into the data stream at the access points 114 or, as in the present embodiment, closely or immediately preceding the access points 114. The index data 110 may be used to quickly identify subsequent access points 114 or subsequent index data to a viewing or editing system accessing the data stream. In the example shown, a block of index data may include pointers multiple access points or multiple groups of pictures. However, in an alternative implementation, the index data may merely point beyond a single group. This alternative has the benefit that the index data can be written as the data is coded only requiring buffering of a single group before the index can be written and the coded data can be efficiently scanned by jumping from index to index without any parsing of intervening data.

A compressed data stream may further comprise layer 1 predicted frames (equivalent to MPEG P-frames) 118, which are coded using references to at least one layer 0 preceding frame and layer 2 frames 116 which are coded with reference to a layer 1 frame (equivalent to MPEG B-frames (bi-directionally-predicted frames), which are coded using references to both preceding and subsequent frames).

In an alternative embodiment, the index data may be inserted into the data stream after encoding of the data stream, for example by parsing the data stream to determine the location of the access points.

The structure of one embodiment of index data is illustrated in more detail in Fig. 34. The configuration of the index data and the data contained within it may vary, but the data preferably includes an offset value indicating the position of the next access point or index data in the data stream, for example by indicating the number of bits to the next access point.

The index data may further contain information relating to the subsequent section of data. For example, the index data may contain some or all of the following data: the number and arrangement of I-, P- and B-frames in the subsequent section of data, the length of the section or data and of each frame, the coding format used for the section of data. Hence each section of index data may comprise an index table, which may contain the index data for the section of data corresponding to that index data.

As discussed above, media players that implement random access to a compressed video stream can be difficult to implement with variable bit rate codecs. MPEG falls into this category because even with "cbr (constant bit rate)" MPEG the frames do not necessarily have a fixed size. To move to a given frame a user has to find the nearest access point in the bit stream. With MPEG this involves some searching and significant parsing of the bit stream. This makes random access difficult. Alternatively the entire stream may be parsed to create an index table that, used in conjunction with the coded bit stream, enables efficient random access. Essentially MPEG was not designed with random access in mind and this has inhibited its take up for file-based applications such as internet streaming and desktop production.

In order to ensure that a codec (coding / decoding technique or scheme) is applicable to as many applications as possible, it is desirable to facilitate the use of the codec for file based applications. Furthermore the search strategies used to achieve semi-scrubbable coding for MPEG are more difficult for arithmetic coding techniques because the use of arithmetic coding makes parsing the bitstream more difficult. It is impractical to build complete index tables into a header in the bitstream; this would preclude its use in streaming applications. However, as decribed above, it is possible to periodically embed partial index tables. At the very least the offset to the next access point should be provided and, probably, a list of frames included in the current access group (i.e. prior to the next access point).

Hence some indexing information may be added to the compressed bitstream to enable random access and scrubbable players without compromising the use of the data in streaming applications.

It will be appreciated that modifications of detail may be made within the scope of the invention. As explained above, features may be provided independently or in other combinations.

CLAIMS

1. A data processing method for processing a stream of data to be coded using prediction, the stream comprising at least first, second and third portions, in which the second portion may be coded using prediction based on the first portion and the third portion may be coded using prediction based on at least the second portion, the method comprising:
forming an estimate of accuracy of a prediction of the second portion based on the first portion; and selectively predicting the third portion based on at least the second portion and the estimate of accuracy of the prediction of the second portion.

2. A method according to Claim 1 wherein the estimate is based only on data that will be available to a decoder which is receiving the signal serially.

3. A method according to Claim 1 or 2 wherein the third portion comprises an individual coefficient.

4. A method according to Claim 1 or 2 wherein the third portion comprises a set of coefficients.

5. A method according to any preceding claim further comprising selectively predicting subsequent portions based on the success of preceding predictions.

6. A method according to Claim 5 wherein the second and third portions of a first selective prediction are used respectively as the first and second portions in a second selective prediction method, to predict a new third portion.

7. A method according to any preceding claim wherein selectively predicting the third portion comprises comparing the magnitude of the second portion to the magnitude of the difference between the second and first portions.

8. A method according to Claim 7 wherein the third portion is predicted only if the second portion is substantially greater than said magnitude of the difference between the second and first portions.

9. A method according to Claim 8 wherein the second portion is required to be at least a given multiple of the said magnitude of the difference between the second and first portions.

10. A method according to Claim 9 wherein said multiple is at least two, preferably at least about four, more preferably about 6.

11. A method according to Claim 8 wherein a scaling function is applied to one of a cost measure of the second portion and a cost measure of the difference between the second portion and a prediction

thereof based on the first portion.

12. A method according to any preceding claim wherein each of the second portion and third portions are predicted based on a plurality of preceding elements.

13. A method according to any of Claims 1 to 11 wherein, if the third portion is predicted, the second portion is used as a prediction.

14. A method according to any preceding claim wherein, following selectively predicting, either the third portion or the difference between the third portion and a prediction thereof is coded.

15. A method according to any preceding claim wherein the data is processed by a hierarchical coding scheme in which data is coded in successive layers, wherein the third portion comprises data from a higher layer and the second portion comprises corresponding data from at least one lower layer.

16. A method according to any preceding claim wherein the selective prediction of the third portion comprises a prediction based on the data preceding the third portion if a scaling function applied to a cost measure of the magnitude of the difference between the second portion and a prediction of the second portion based on the data preceding the second portion is less than a cost measure of the magnitude of the second portion, and a default prediction otherwise.

17. A method according to Claim 16 wherein, where the third portion comprises one or more numerical coefficients, the default prediction is zero.

18. A method according to Claim 16 or 17 wherein the scaling function comprises multiplication by a parameter and preferably wherein the parameter has a value of at least two.

19. A method according to Claim 18 wherein the parameter is set to a fixed value.

20. A method according to Claim 18 wherein the value of the parameter is adjusted based on dynamic analysis of the data.

21. A method according to Claim 20 wherein the parameter is adjusted based on at least one of:-

a) The prior inputs to the selective prediction method;
b) The prior outputs of the selective prediction method;
c) The prior outputs of the prediction based on the data preceding the portion to be coded ;
d) The prior outputs of the comparison.

22. A method according to any preceding claim wherein the method of prediction is dynamically

adjusted based on the results of prediction for preceding data.

23. A method according to Claim 22 wherein the effectiveness of the selective prediction method is compared with the effectiveness of the basic prediction method.

24. A method of selectively predicting a current portion of data based on preceding data comprising selectively predicting a current portion of data x(i) in accordance with the formula:

$$P_1(S(i-1)) = \begin{cases} P(S(i-1)) \text{ if } \lambda \cdot C(P(S(i-2)) - x(i-1)) < C(x(i-1)) \\ 0 \text{ otherwise} \end{cases}$$

wherein

P₁(S(i-1)) represents the selective prediction of the current portion of data based on at least a part of the set of data preceding the current portion;

P(S(i-1)) represents a prediction of the current portion of data based on at least a part of the set of data preceding the current portion;

P(S(i-2)) represents a prediction of the preceding portion of data based on at least a part of the set of data preceding the preceding portion;

C represents a measure of cost of transmitting or storing the data;

λ represents a parameter;

x(i-1) represents the preceding portion of data.

25. A method according to Claim 24 wherein each portion of data comprises a single coefficient.

26. A method according to Claim 24 wherein each portion of data comprises a group of coefficients.

27. A method according to Claim 25 wherein each prediction comprises the preceding coefficient.

28. A method according to Claim 24 or 25 wherein the cost measure comprises the magnitude of the coefficient.

29. A method according to Claim 21a, 24 or 25 wherein the cost measure comprises a measure of the entropy of the coefficient or coefficients.

30. A method according to any of Claims 24 to 29 wherein at least one of the cost measure, the method of prediction and the parameter are adapted in response to successive portions of data.

31. A method according to any preceding claim wherein portions to be coded are quantised prior to prediction.

32. A method according to any of Claims 1 to 30 wherein portions to be coded are unquantised and the residual data after selective prediction is quantised.

33. A method according to Claim 32 wherein the coefficients used for prediction are reconstructed coefficients, on which inverse quantisation and prediction have been performed.

34. A method according to Claim 32 or 33 wherein, in the case of adaptive modification of prediction method or cost measure or prediction parameters, adjustment is based on reconstructed coefficients.

35. A method of coding a video picture in which the picture is partitioned into at least first and second portions, the method comprising predicting elements of the second portion by displacing elements of the first portion and coding the differences between elements of the second portion and the respective predictions thereof.

36. A method according to Claim 35 wherein displacing comprises displacing substantially diagonal edges to be more aligned vertically or horizontally.

37. A method according to Claim 35 or 36 wherein coding comprises coding by a wavelet coding method.

38. A method according to Claim 35 wherein the first and second portions correspond respectively to first and second times.

39. A method according to Claim 38 wherein displacing elements comprises displacing elements based on estimated motion between the first and second times.

40. A method according to any of Claims 35 to 39 wherein elements of the first portion are positioned at spatially different but substantially adjacent positions to elements of the second portion.

41. A method according to Claim 40 wherein the first and second portions are interleaved.

42. A method according to Claim 41 wherein the portions comprise alternating picture lines.

43. A method according to Claim 41 wherein the first and second portions comprise fields of an interlaced video picture frame.

44. A method of coding an interlaced video picture frame comprising at least first and second fields, each field comprising a plurality of picture lines, the lines of the first field interleaving with lines of the second field, the first and second fields corresponding to respective first and second times, the method comprising:

predicting elements of the second field based on the first field, wherein predicting comprises shifting elements of the first field along the direction of said picture lines based on an estimated component of motion along said lines between said first and second times, whereby elements of the second field are predicted based on elements of the first field which are estimated to be aligned;

coding the second field following prediction.

45. A method according to Claim 42 or 44 wherein the lines are horizontal and only horizontal motion is estimated.

46. A method according to Claim 42 or 44 wherein motion is estimated in two directions.

47. A method according to Claim 46 wherein two-dimensional motion vectors are determined.

48. A method according to Claim 46 wherein motion is estimated separately for the two directions.

49. A method according to any of Claims 35 to 48 wherein a single element of the first portion or field is used to predict each element of the second portion or field.

50. A method according to any of Claims 35 to 48 wherein a plurality of elements of the first portion or field is used to predict each element of the second portion or field.

51. A method of coding an interlaced video picture frame comprising at least first and second fields, each field comprising a plurality of picture lines, the lines of the first field interleaving with lines of the second field, the first and second fields corresponding to respective first and second times, the method comprising:

predicting elements of the second field based on the first field, wherein predicting comprises shifting elements of the first field along the direction of said picture lines but not perpendicular to said lines based on an estimated component of motion along said lines between said first and second times, whereby elements of the second field are predicted based on elements of the first field which are estimated to be aligned in one direction only;

coding the second field following prediction.

52. A method according to any of Claims 35 to 51 wherein elements of the first portion or field are used both as a basis for estimating a component of apparent motion between the first and second portions or fields and as a basis for predicting values of elements of the second portion or field.

53. A method of coding an interlaced video frame comprising first and second fields, the method comprising predicting elements of the second field based on elements of the first field which are

displaced horizontally in accordance with estimated horizontal components of motion between the first and second fields and coding the second field following prediction.

54. A method according to any of Claims 43 to 53 wherein only a single estimate of horizontal motion is used for each picture line.

55. A method according to any of Claims 43 to 54 wherein each element of the second field is associated with a predictor comprising an element of the first field from a corresponding picture line at a position along the corresponding line corresponding to the position along the line of the element of the second field to be predicted plus an offset based on estimated motion.

56. A method according to any of Claims 43 to 55 wherein the picture is coded by a wavelet coding method following prediction.

57. A method according to any of Claims 43 to 56 wherein a difference between each element of the second field and a corresponding element of the first field is coded.

58. A method according to Claim 57 wherein the corresponding element of the first field comprises an element of the first field shifted based on a component of estimated motion.

59. A method according to Claim 57 or 58 wherein said difference is used to provide a high-pass output of a wavelet coding filter, preferably wherein half the difference is coded.

60. A method according to Claim 59 wherein a low pass output of the wavelet coding filter comprises the corresponding a component if the estimated prediction predicts no elements on a line or, if one or more elements on the second line are predicted, the average of the corresponding element of the first field and the predicted elements of the second field.

61. A method according to any of Claims 35 to 60 wherein each element comprises an individual pixel.

62. A method according to any of Claims 35 to 60 wherein each element comprises a group of pixels.

63. A method according to any of Claims 35 to 62 wherein the frame is coded without further motion estimation.

64. A method according to Claim 63 and 56 wherein further wavelet decomposition is performed without further motion estimation.

65. A method according to Claim 1 wherein the data is coded by a hierarchical coding method having

at least parent and child coding levels and wherein third portion comprises a portion of the child coding level and the second portion comprises a portion of the parent coding level.

66. A method of hierarchical coding of data in which data is coded by a coding scheme into at least parent and child coding levels, wherein parent and child levels are both quantised, the method comprising selecting quantisation parameters for at least a portion of the child coding level based on the quantisation parameters for at least a corresponding portion of the parent coding level.

67. A method according to Claim 66 wherein the coefficients in the parent coding level are partitioned and wherein the partitioning of the child level is based on the partitioning of the parent level.

68. A method according to Claim 67 wherein partitioning of the parent level is based on the quantised coefficients of the parent level.

69. A method according to Claim 68 wherein the parent level is communicated to a decoder prior to the child level.

70. A method of decoding a signal coded by a method according to any preceding claim comprising reconstructng a portion of the signal to be reconstructed based on a received portion of data and a method of selective prediction of a subsequent portion corresponding to said method according to any preceding claim.

71. A method according to Claim 70 of decoding a signal coded according to Claim 1, the method comprising receiving and decoding first and second coded portions of data, selectively predicting a third portion of data based on the first and second portions and applying received coded data for the third portion to the selective prediction to reconstruct the third portion.

72. A method according to Claim 70 of decoding data coded hierarchically according to Claim 66, the method comprising:

receiving data encoding a parent level and reconstructing the parent level;

reconstructing the child level based on received data for the child level and re-using data used in the reconstruction of the parent level.

73. A method of coding a sequence of pictures comprising:

selecting only a subset of the sequence of pictures for communication leaving a subset of residual pictures;

interpolating from the subset of pictures for communication according to an interpolation algorithm to create at least one interpolated picture corresponding substantially to one of the subset of residual pictures;

encoding adjustment information based on a difference between the at least one interpolated picture and the corresponding residual picture;
communicating the subset of pictures for communication together with the adjustment information.

74. A method according to Claim 73 wherein the subset of pictures is encoded.

75. A method according to Claim 73 or 74 wherein the sequence of pictures has a first frame rate and wherein the subset of pictures comprises a sequence of pictures having a lower frame rate than the first frame rate.

76. A method according to Claim 75 wherein pictures are dropped as residual pictures at substantially regular intervals to reduce the frame rate.

77. A method according to Claim 75 or 76 wherein pictures are dropped as residual pictures based on picture information content.

78. A method according to any of Claims 73 to 77 wherein pictures are dropped as residual to control the output frame rate.

79. A method according to any of Claims 73 to 78 wherein pictures are dropped as residual to control the output bitrate.

80. A method according to any of Claims 73 to 79 wherein both selection of pictures and coding parameters are adjusted to control the output bitrate.

81. A method according to any of Claims 73 to 80 further comprising communicating the output to be received by a decoder arranged to perform interpolation in accordance with the interpolation algorithm.

82. A method of reconstructing a sequence of pictures comprising:

receiving a sequence of pictures to produce a sequence of received pictures;

interpolating at least one picture from the received pictures according to a predetermined interpolation algorithm;

receiving adjustment information and applying the adjustment information to the or each interpolated picture to modify the or each interpolated picture;

outputting a sequence of pictures comprising the received pictures and the modified interpolated pictures.

83. A method according to Claim 82 including decoding the sequence of pictures received to provide received pictures.

84. A method of communicating a sequence of pictures comprising:
at a coder,
selecting only a subset of the sequence of pictures for communication leaving a subset of residual pictures;
interpolating from the subset of pictures for communication according to an interpolation algorithm to create at least one interpolated picture corresponding substantially to one of the subset of residual pictures;
encoding adjustment information based on a difference between the at least one interpolated picture and the corresponding residual picture;
communicating the subset of pictures to a decoder together with the adjustment information; and
at the decoder,

receiving the subset of pictures to produce a sequence of received pictures;

interpolating at least one picture from the received pictures according to the interpolation algorithm;

receiving the adjustment information and applying the adjustment information to the or each interpolated picture to modify the or each interpolated picture;

outputting a sequence of pictures comprising the received pictures and the modified interpolated pictures.

85. A method according to any of Claims 73 to 84 wherein the interpolation algorithm comprises interpolating based only on information available for one or more pictures other than the picture to be interpolated.

86. A method according to Claim 85 wherein the pictures other than the picture to be interpolated comprise pictures encoded using prediction based on motion vectors.

87. A method according to Claim 86 wherein the motion vectors for the other pictures are used in interpolation.

88. A method according to Claim 87 wherein motion vectors are communicated for pictures selected for communication but wherein no motion vectors are communicated specifically for the interpolated pictures.

89. A method according to any of Claims 73 to 88 wherein the adjustment information includes information concerning the timing of an interpolated frame to which the adjustment information relates.

90. A method according to any of Claims 73 to 88 wherein adjustment information is provided for all pictures to be interpolated.

91. A method according to any of Claims 73 to 88 wherein pictures may be interpolated without adjustment.

92. A computer program or computer program product for performing a method according to any preceding claim.

93. A coder arranged to perform a method according to any of Claims 1 to 69 or 73 to 81.

94. A decoder arranged to perform a method according to any of Claims 70 to 72 or 82 to 83.

95. Apparatus substantially as any one herein described or as illustrated in the accompanying drawings.

96. A method substantially as any one herein described, with reference to the accompanying drawings.

97. A coder arranged to process a stream of data to be coded using prediction, the stream comprising at least first, second and third portions, in which the second portion may be coded using prediction based on the first portion and the third portion may be coded using prediction based on at least the second portion, the coder comprising:
means for forming an estimate of accuracy of a prediction of the second portion based on the first portion; and
means for selectively predicting the third portion based on at least the second portion and the estimate of accuracy of the prediction of the second portion.

98. A coder arranged to process a stream of data to be coded using prediction, the stream comprising at least first, second and third portions, in which the second portion may be coded using prediction based on the first portion and the third portion may be coded using prediction based on at least the second portion, the coder comprising:
a prediction estimator arranged to form an estimate of accuracy of a prediction of the second portion based on the first portion; and
a selective predictor arranged to selectively predict the third portion based on at least the second portion and the estimate of accuracy of the prediction of the second portion.

99. A coder for coding a video picture in which the picture is partitioned into at least first and second portions, the coder comprising means for predicting elements of the second portion by displacing elements of the first portion and means for coding the differences between elements of the second portion and the respective predictions thereof.

100. A coder for coding an interlaced video picture frame comprising at least first and second fields, each field comprising a plurality of picture lines, the lines of the first field interleaving with lines of the second field, the first and second fields corresponding to respective first and second times, the coder comprising:

means for predicting elements of the second field based on the first field, wherein predicting comprises shifting elements of the first field along the direction of said picture lines based on an estimated component of motion along said lines between said first and second times, whereby elements of the second field are predicted based on elements of the first field which are estimated to be aligned in said direction;

means for coding the second field following prediction.

101. A coder for coding an interlaced video picture frame comprising at least first and second fields, each field comprising a plurality of picture lines, the lines of the first field interleaving with lines of the second field, the first and second fields corresponding to respective first and second times, the coder comprising:

a predictor for predicting elements of the second field based on the first field, wherein predicting comprises shifting elements of the first field along the direction of said picture lines based on an estimated component of motion along said lines between said first and second times, whereby elements of the second field are predicted based on elements of the first field which are estimated to be aligned in said direction;

a coding module for coding the second field based on the output of said predictor.

102. A coder for coding a sequence of pictures comprising:

means for selecting only a subset of the sequence of pictures for communication leaving a subset of residual pictures;

an interpolator for interpolating from the subset of pictures for communication according to an interpolation algorithm to create at least one interpolated picture corresponding substantially to one of the subset of residual pictures;

means for encoding adjustment information based on a difference between the at least one interpolated picture and the corresponding residual picture;

means for communicating the subset of pictures for communication together with the adjustment information.

103. A coder according to Claim 102 including means for encoding the subset of pictures for communication.

104. A method of processing a media data stream comprising:

providing a compressed media data stream having a plurality of access points, wherein the access points comprise points from which media playback can be commenced;

identifying access points in the media data stream;
inserting index data at a point in the media data stream to identify at least one subsequent access point.

105. A method according to Claim 104 wherein the step of providing a compressed media data stream comprises receiving a compressed media data stream.

106. A method according to Claim 104 wherein the step of providing a compressed media data stream comprises receiving a media data stream and compressing the media data stream.

107. A method according to Claim 106 wherein the steps of identifying access points and inserting index data are performed in conjunction with the step of compressing the media data stream.

108. A method according to any of Claims 104 to 107 wherein the media data stream comprises a video stream comprising a plurality of frames.

109. A method according to any of Claims 104 to 108 wherein the index data comprises at least one index table.

110. A method according to Claim 109 wherein each index table comprises an offset value indicating the position of the next access point or index data within the data stream.

111. A method according to any of Claims 104 to 110 wherein the index data comprises an indication of the structure of at least some frames closely, preferably immediately, following the index data.

112. A method according to any of Claims 104 to 111 wherein the access points comprise elements in the data stream that have been compressed substantially without reference to other elements in the data stream.

113. A method according to Claim 112, wherein the media data stream comprises a video stream and wherein elements in the data stream comprise frames.

114. A method according to any of Claims 104 to 113 wherein the access points comprise I-frames in an MPEG-encoded data stream.

115. A method, preferably according to any of Claims 1 to 91, wherein a global motion component is optionally transmitted for each frame.

116. A method according to Claim 115 wherein both global motion and block motion are optionally transmitted.

117. A method according to Claim 115 or 116 wherein a flag indicating whether a default, preferably zero, global motion component is to be used.

118. A method according to any of Claims 115 to 117 wherein block motion is transmitted as differential motion with respect to the global motion.

119. A method according to any of Claims 115 to 118 wherein the global motion is encoded as a description of an affine transformation.

120. A method of processing a series of frames to be coded using prediction, the series comprising at least a first and a second frame, in which the second frame may be coded using prediction based on the first frame, the method comprising:
deriving a global motion component for predicting the second frame based on the first frame;
deriving block motion components for predicting elements of the second frame based on respective corresponding elements of the first frame;
coding the second frame based on the global motion component and the block motion components.

121. A method according to Claim 120 wherein the block motion components are coded based on the difference between a local estimate of motion for each element and the motion of the element predicted by the global motion component.

122. A method according to Claim 120 or 121 wherein a default global motion component is selectively used in place of a derived global motion component.

123. A method according to Claim 122 wherein only one default global motion component is defined, preferably zero motion.

124. A method according to Claim 123 wherein a flag is encoded to signify that a default global motion is used.

125. A method according to any of Claims 120 to 125 wherein the global motion component is encoded as information identifying an affine transformation.

126. A method according to any of Claims 120 to 125 wherein the global motion component is encoded as information identifying at least one of:
a translational motion component;
a rotational motion component; and
a resizing motion component.

127. A method according to any of Claims 120 to 126 wherein displacements based on the global motion are calculated for at least first and second blocks, wherein the displacement for the second block is calculated by adding a component based on a transformation corresponding to the global motion to the displacement for the first block.

128. A method according to Claim 127 wherein the component based on the transformation is scaled based on the displacement between the first and second blocks.

129. A method according to Claim 128 wherein the block spacing is constant and wherein a constant corresponding to the scaled component based on transformation is stored for adding to the displacement of the first block to calculate the displacement of the second block and for adding to the displacement of the second block to calculate the displacement of a third block.

130. A method according to any of Claims 127 to 129 wherein, following initial determination of the scaled component based on the transformation corresponding to global motion, only one addition for each of the components of the displacement are performed to determine the displacement for at least two blocks, preferably wherein only two additions are performed for x and y components.

131. A method according to any of Claims 115 to 130 wherein information is transmitted signifying a selected one of at least two of the following coding conditions:-
zero motion is used;
only global motion is used;
global motion and block motion are both used.

132. A method according to Claim 131 wherein block motion is coded differentially, preferably as adjustments to the global motion.

133. A method according to Claim 131 wherein global motion, block motion and motion compensated transform residuals are encoded.

134. A method according to Claim 133 wherein the transform is a wavelet transform.

135. A method according to any of Claims 115 to 134 wherein information is encoded signifying no more data for the current frame.

136. A method of coding a sequence of pictures wherein a plurality of coding decisions are dynamically made between pictures, the method comprising encoding a flag signifying no further data for the current picture to enable a receiver to determine at least one coding decision in the absence of specific information concerning the coding decision.

137. A method of decoding a sequence of pictures wherein a plurality of coding decisions are dynamically made between pictures, the method comprising seeking a flag signifying no further data for the current picture and determining at least one coding decision based on receipt of the flag in the absence of specific information concerning the coding decision.

138. Use of a flag signalling no further data for a picture in the coding of a sequence of pictures to convey information concerning at least one selective coding decision concerning a picture.

139. A method or use according to any of Claims 136 to 138 wherein the flag signifying no further data for the current picture is used to signify selection between a plurality of levels of coding complexity.

140. A method or use according to any of Claims 136 to 139 wherein the flag signifying no further data for the current picture is used to signify that a picture or portion of a picture is omitted from coding.

141. A method according to Claim 140 wherein the omitted picture or portion is to be derived at a decoder.

142. A method according to Claim 141 wherein the omitted picture or portion is derived by interpolation.

143. A method of coding a sequence of pictures comprising selectively omitting pictures wherein information signifying omission of pictures is selectively encoded in place of omitted pictures.

144. A method of decoding a sequence of pictures comprising selectively omitted pictures, the method comprising receiving information signifying omission of pictures and decoding the sequence including substituting synthetic picture data in place of omitted pictures.

145. A method according to Claim 144 wherein synthetic picture data is obtained by interpolating pictures.

146. A method of coding a series of interlaced video picture frames, wherein each frame comprises at least first and second fields, each field comprising a plurality of picture lines, the lines of the first field interleaving with lines of the second field, the first and second fields corresponding to respective first and second times and having respective first and second parities, the method comprising:
predicting elements of the first field of a frame based on a weighted combination of at least one field of a preceding or subsequent frame in the series of interlaced video picture frames having the same parity as the first field and at least one field of a preceding or subsequent frame in the series of interlaced video picture frames having the opposite parity to the first field; and
coding the first field following prediction.

147. A method according to Claim 146 wherein the method further comprises:
predicting elements of the second field of a frame based on a weighted combination of at least one field of a preceding or subsequent frame in the series of interlaced video picture frames having the same parity as the second field and at least one field of a preceding or subsequent frame in the series of interlaced video picture frames having the opposite parity to the second field; and
coding the second field following prediction.

148. A method according to Claim 146 or 147 wherein a weighted combination is determined based on the number of field lines traversed by the element between frames.

149. A method according to Claim 148 wherein the weighted combination comprises substantially zero contribution from the opposite parity field when the number of field lines traversed is even, substantially full contribution from the opposite parity field when the number of field lines traversed is

odd and a combination of fields of both parities in the case that the number of field lines traversed is non integral.

150. A method according to any of Claims 146 to 149 wherein the weighted combination applies non-integer value weights in the case of non-integral number of lines traversed.

151 A method according to any of Claims 146 to 149 wherein the weighted combination is determined based on at least one of the least significant bit of the vertical motion of the element and the fractional part of the vertical motion of the element.

152. A method according to any of Claims 146 to 151 wherein the step of predicting an element of the first or the second field comprises deriving a motion vector for the element.

153. A method according to any of Claims 146 to 152 wherein the picture is coded by a wavelet coding method following prediction.

154. A method according to any of Claims 146 to 153 wherein each element comprises an individual pixel.

155. A method according to any of Claims 146 to 153 wherein each element comprises a group of pixels.

156. A method according to Claim 155 wherein the group of pixels comprises a block in the first or second field.

157. A method according to any of Claims 146 to 156 wherein the lines are horizontal and only vertical motion is estimated.

158. A method according to any of Claims 146 to 156 wherein motion is estimated in two directions.

159. A method according to any of Claims 1 to 91 wherein the number of coding levels is permitted to vary dynamically within a sequence of pictures.

160. A method of coding a sequence of pictures comprising using a hierarchical transformation with a plurality of coding levels to code the sequence characterised in that the number of coding levels is selectively made to vary dynamically during coding.

161. A method according to Claim 159 or 160 wherein the number of coding levels varies from picture to picture.

162. A method according toClaim 159 or 160 wherein the number of coding levels varies within a picture.

163. A method according to any of Claims 159 to 162 wherein the coding levels comprise levels of a wavelet transform.

164. A method according to any of Claims 1 to 91 wherein the block size is permitted to vary within a sequence.

165. A method of coding a sequence of pictures comprising coding each picture as a plurality of blocks characterised in that the block size is permitted to vary dynamically during coding.

166. A method according to Claim 164 or 165 wherein the block size varies from picture to picture.

167. A method according to Claim 164 or 165 wherein the block size varies within a picture.

168. A method according to any of Claims 164 to 167 further comprising signalling block size, or a change in block size, for a picture or picture portion.

169. A method according to any of Claims 1 to 91 wherein motion vector precision is permitted to vary within a sequence.

170. A method of coding a sequence of pictures comprising coding each picture based on prediction using motion vectors characterised in that motion vector precision is permitted to vary within a sequence dynamically during coding.

171. A method according to Claim 169 or 170 wherein the motion vector precision varies from picture to picture.

172. A method according to Claim 169 or 170 wherein the motion vector precision varies within a picture.

173. A method according to any of Claims 169 to 172 further comprising signalling motion vector precision, or a change in motion vector precision, for a picture or picture portion.

174. A method of coding a sequence of pictures, preferably according to any of Claims 1 to 91, wherein pictures are coded by prediction based on one or more reference pictures, characterised in that the reference pictures are selectively chosen during coding.

175. A method according to Claim 174 wherein the reference pictures to be used to code a given frame are selected from a buffer of reference pictures.

176. A method according to Claim 175 wherein the buffer contains at least 4 reference pictures.

177. A method according to Claim 175 wherein the buffer contains exactly 4 reference pictures

178. A method according to any of Claims 174 to 177 wherein the reference pictures are permitted to include a "long-term" reference picture.

179. A method according to any of Claims 174 to 177 wherein the reference pictures are permitted to include a reference frame which is not displayed.

180. A method according to any of Claims 174 to 179 the number of reference pictures used to predict a picture is limited to two.

181. A method of coding a picture, preferably according to any of claims 1 to 91, wherein information relating to the picture is quantised characterised in that the number of quantisation levels is arranged to vary across a picture or corresponding subband.

182. A method according to Claim 181 wherein at least one region of greater interest than another region is defined.

183. A method according to Claim 182 wherein the region of greater interest corresponds to a central picture region and the other region corresponds to an edge picture region, wherein more quantisation levels are used in the region of greater interest.

184. A method according to Claim 181 or 182 or 183 wherein a plurality of discrete numbers of quantisation levels is provided.

185. A method according to Claim 181 or 182 or 183 wherein the number of qunatisation levels is arranged to vary substantially gradually across the picture.

186. A method according to Claim 185 wherein a weighting function comprising variable weighting factors is used in the derivation of quantisation levels for a picture or corresponding subband.

187. A method according to any of Claims 1 to 91 wherein a non-linear quantisation scheme is used.

188. A method of coding data, preferably a sequence of pictures, preferably according to any of Claims 1 to 91, wherein parent-child relationships are defined between components of the data, the method comprising coding the data and quantising the data, characterised in that a first quantiser is used for the case where parent has a zero coefficient and a second quantiser is used for the case where the parent has a non-zero coefficient.

189. A method according to Claim 188 wherein different quantisers and contexts are used in different regions of a single picture.

190. A method of coding a sequence of pictures comprising coding the pictures and quantising the coded pictures, characterized by using a first quantiser and context in a first picture region and a second quantiser and context in a second picture region.

191. A method of coding data, preferably data for a sequence of pictures, preferably according to any of Claims 1 to 91, the method comprising coding data to yield symbols and binary coding the symbols according to a binarisation scheme, characterised by dynamically selecting a binarisation scheme from among a plurality of available binarisation schemes based on the properties of the data.

192. A method according to Claim 191 wherein a feature of the coded data is compared to a threshold.

193. A method according to Claim 192 wherein the data comprises wavelet coded pictures and wherein parent residual wavelet coefficients are compared to a threshold, preferably 4.

194. A method according to any of Claims 191 to 193 wherein at least one of the available binarisation schemes is selected from the group consisting of unary coding and exp-Golomb coding.

195. A method according to any of Claims 191 to 194 wherein an active binarisation scheme or change in scheme is signalled implicitly based on a property of the data.

196. A method of coding data, preferably a sequence of pictures, preferably according to any of Claims 1 to 91, wherein adaptive arithmetic coding is used, the method comprising coding data using a statistical context and periodically refreshing the context, characterised in that on refreshing the context the statistical context is initialised to a state based on the previous history of the data in place of an initial default value.

197. A method according to Claim 196 further comprising signalling the initial state to enable a decoder to refresh to the same initial state.

198. A method according to Claim 196 wherein optionally the context is refreshed to a default state.

199. A method according to Claim 198 further comprising signalling whether to revert to a default state or a historical initial state to enable a decoder to refresh to the same initial state.

200. A method of arithmetic coding a plurality of symbols according to a coding scheme, preferably according to Claim 196, the method comprising coding the symbols according to the coding scheme, updating statistical contexts as symbols are coded and periodically refreshing contexts, characterised in that, following a reset of statistical context, the probabilities of the symbols are set to values based on the previously coded data.

201. A method of coding data, preferably a sequence of pictures, preferably according to any of Claims 1 to 91, the method comprising coding the data to yield first and second coded data types and arithmetically coding the first and second data according to a statistical context, characterised in that the first data type is coded according to a first statistical context and the second data type is coded according to a second statistical context.

202. A method according to Claim 201 wherein the first data type comprises an intra coded frame and the second data type comprises an inter-coded frame.

203. A method according to any of Claims 1 to 91 wherein a motion vector is estimated based on first and second pictures and wherein the motion vector is projected into at least one third field.

204. A method according to Claim 203 wherein the motion vector is an initial motion vector and wherein a refined motion vector is derived following the projecting.

205. A method of coding a sequence of pictures wherein pictures are coded by prediction based on one or more reference pictures, characterised in that the reference pictures are selectively chosen during coding and that a weighting factor is applied to the or each reference picture during coding.

206. A method according to any of Claims 174 to 180 or Claim 205 further comprising applying a weighting factor to at least one reference picture before coding the picture based on the reference picture.

207. A method according to Claim 206 wherein the weighting factor for the reference picture varies between the pictures being coded.

208. A method according to Claim 206 or 207 wherein the weighting factor for the reference picture

varies based on the temporal displacement of the picture being coded from the reference picture.

209. A method according to any Claims 206 to 208 wherein the weighting factor for the reference picture varies based on the position of the reference picture in the sequence of pictures relative to the picture being coded.

210. A method according to any of Claims 206 to 209 wherein the weighting factor is varied to accommodate a fade effect in a series of pictures.

211. A method according to any of Claims 206 to 210 wherein the weighting factor is varied to accommodate an effect to simulate a change in the lighting conditions in a series of pictures.

212. A method according to any of Claims 174 to 180 or 205 to 211 wherein the reference pictures and the pictures being coded comprise frames in a stream of video data.

213. A method according to any of Claims 174 to 180 or 205 to 211 wherein the reference pictures and the pictures being coded comprise fields in a stream of video data.

214. A method according to any of Claims 206 to 213 wherein the weighting factor is calculated for each picture before the picture is coded.

215. A method according to any of Claims 206 to 214 wherein the weighting factor is applied to each pixel of the reference frame.

216. A method according to any of Claims 206 to 215 wherein the weighting factor comprises one or two scaling factors

217. A method according to any of Claims 206 to 216 further comprising providing a flag to indicate the use of default or variable weightings.

218. A method according to any of Claims 174 to 180 or Claims 205 to 217 further comprising using motion vector tracing to code the motion of elements of the picture between a first and a second picture.

219. A method of minimising the error in motion vector tracing in a series of pictures comprising

calculating the error in the quantisation of motion vectors to integer values, determining the total error over a plurality of quantisation operations and correcting the motion vector tracing based on the total error determined.

220. A method according to Claim 219 wherein the error comprises an unquantised motion error.

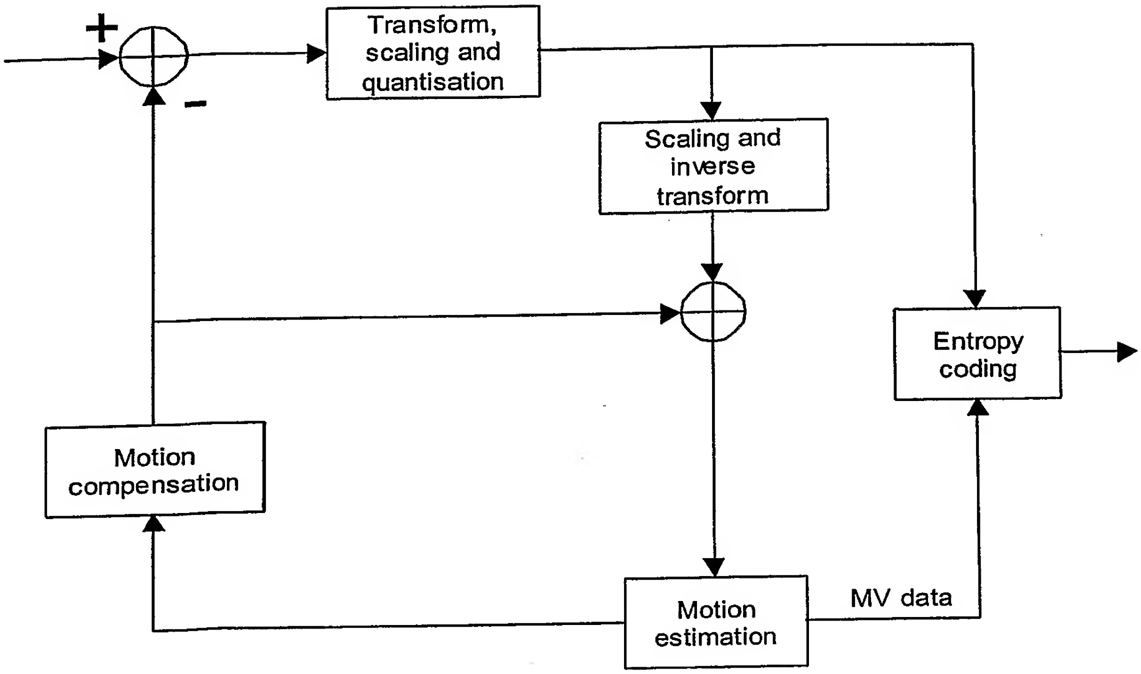

+
−
Transform, scaling and quantisation
Scaling and inverse transform
Entropy coding
Motion compensation
Motion estimation
MV data

Fig. 1

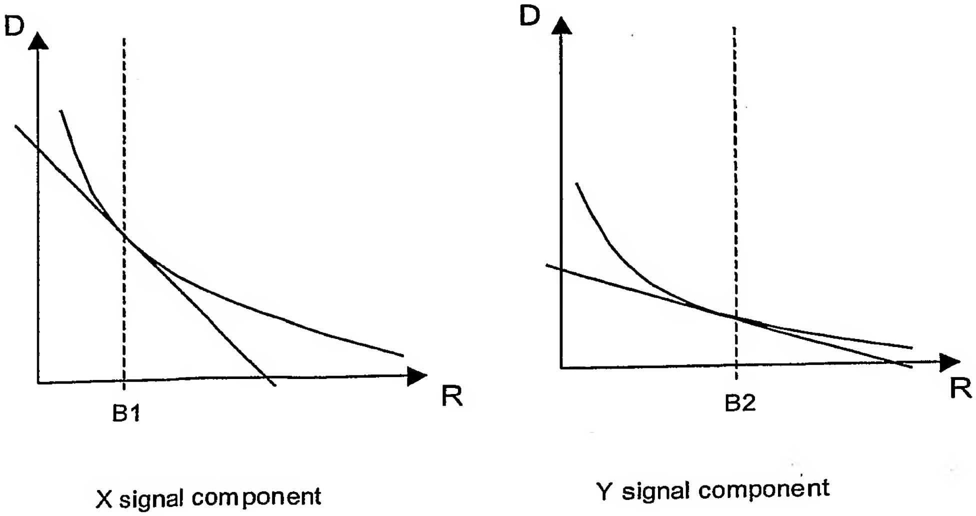
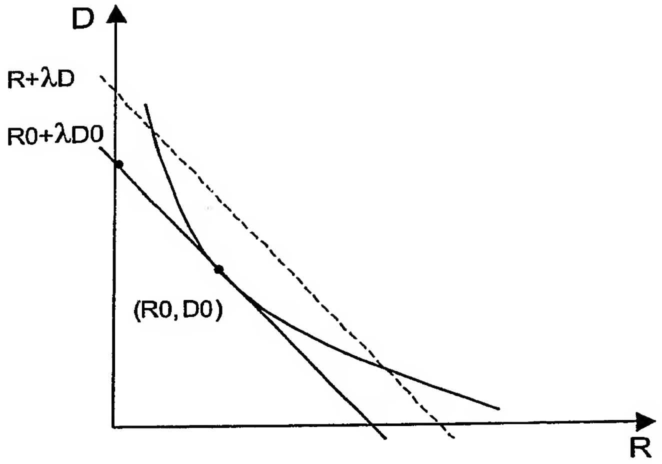

D
R
B1
X signal component
D
R
B2
Y signal component

Fig. 2



Rate-distortion curve

Fig. 3

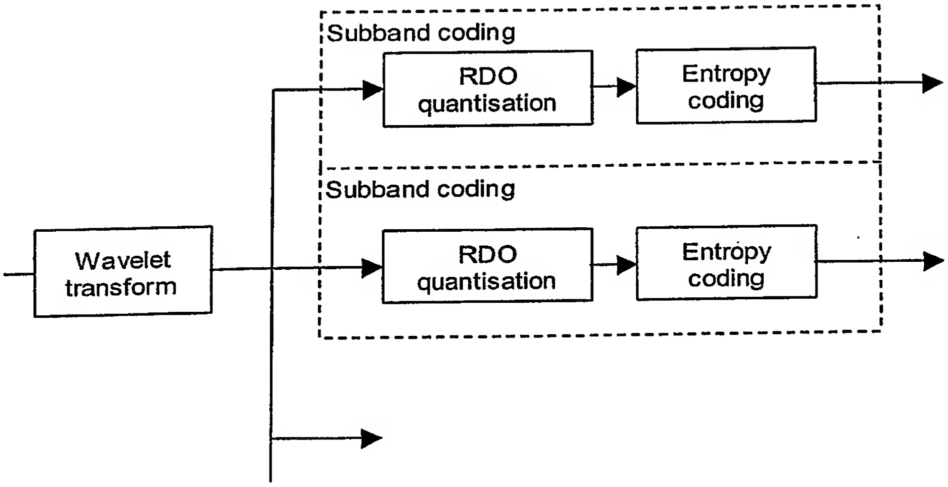


Fig. 4

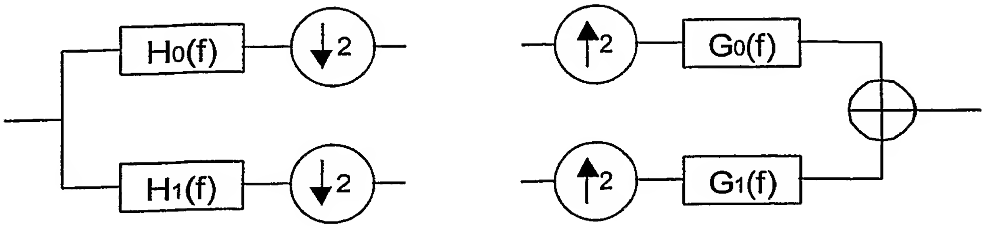


Fig. 5

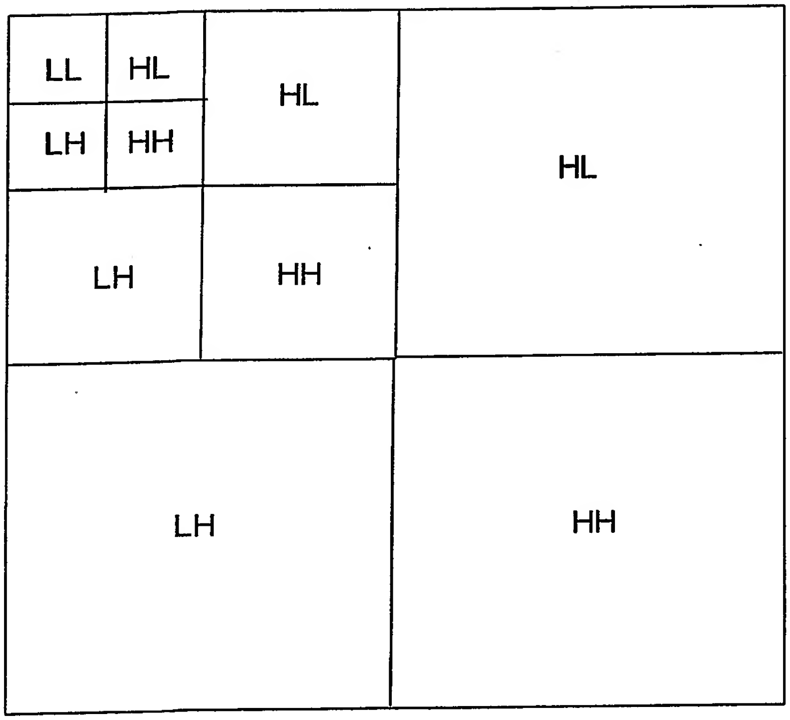


Fig. 6

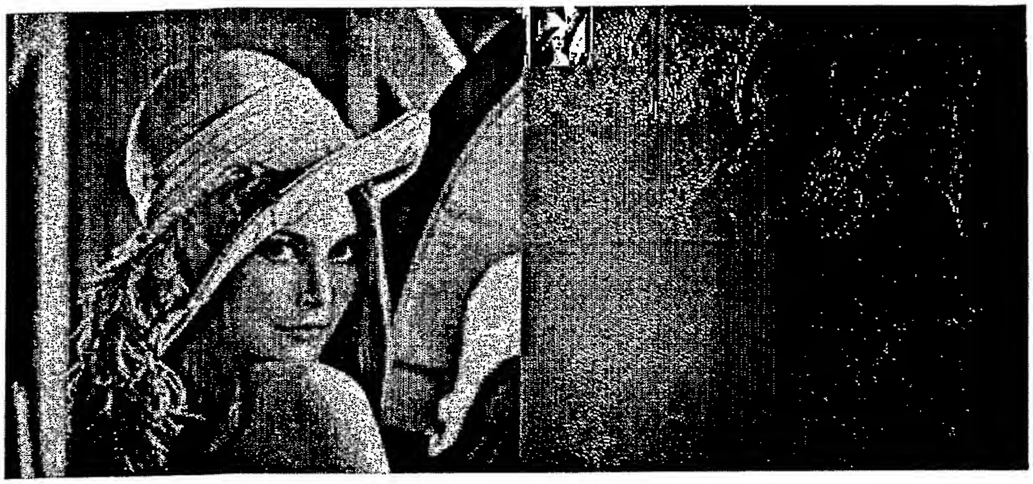


Fig. 7

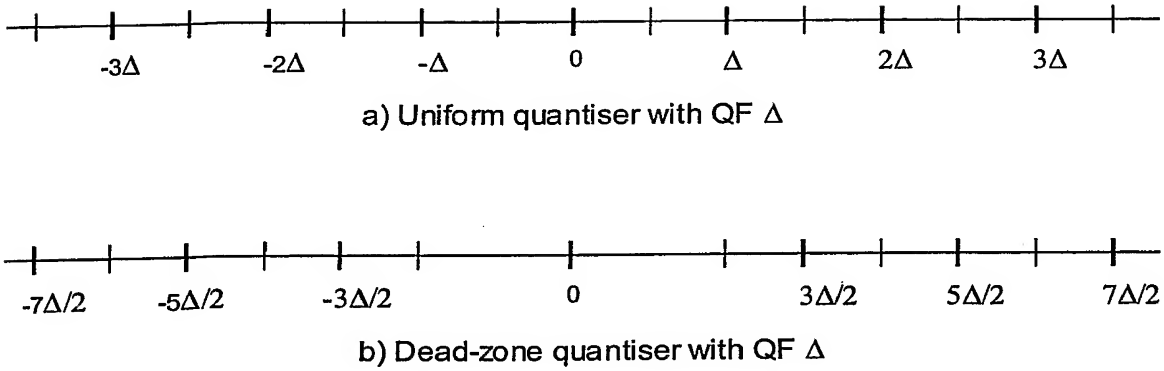

-3Δ
-2Δ
-Δ
0
Δ
2Δ
3Δ
a) Uniform quantiser with QF Δ
-7Δ/2
-5Δ/2
-3Δ/2
0
3Δ/2
5Δ/2
7Δ/2
b) Dead-zone quantiser with QF Δ

Fig. 8

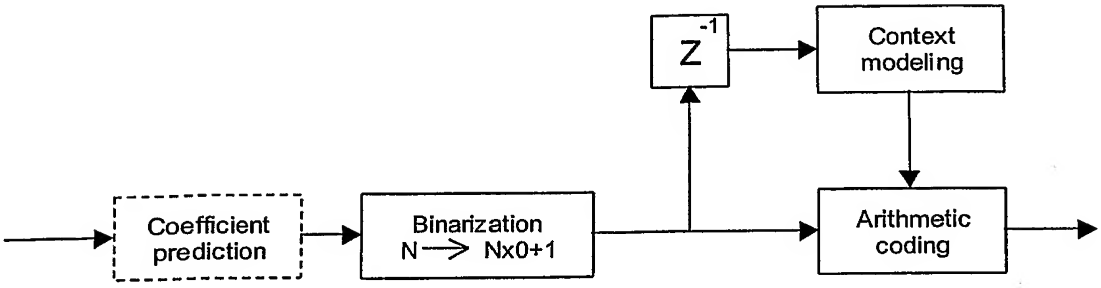


Fig. 9

U(0)	=	1						
U(1)	=	0	1					
U(2)	=	0	0	1				
U(3)	=	0	0	0	1			
U(4)	=	0	0	0	0	1		
U(5)	=	0	0	0	0	0	1	
U(6)	=	0	0	0	0	0	0	1
Bins:		Bin1	Bin2	Bin3	Bin4	Bin5	Bin6	Bin7

Fig.10

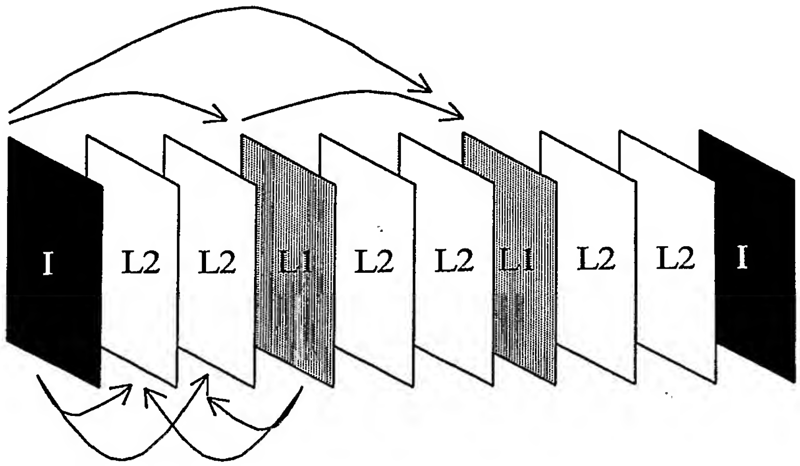


Fig. 11

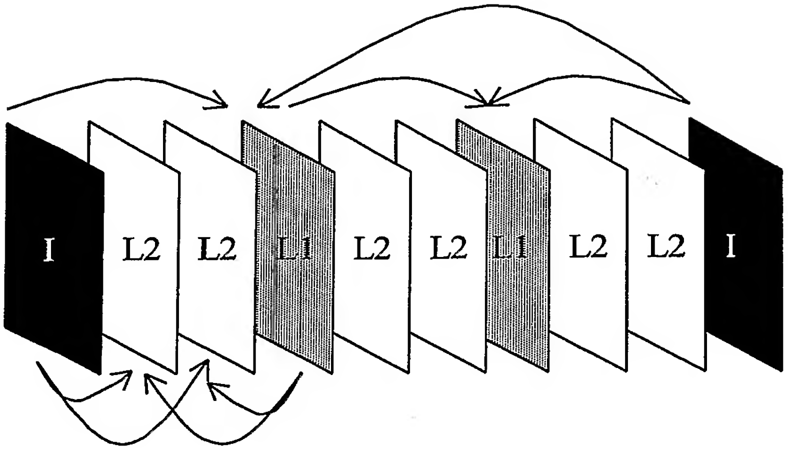


Fig. 12

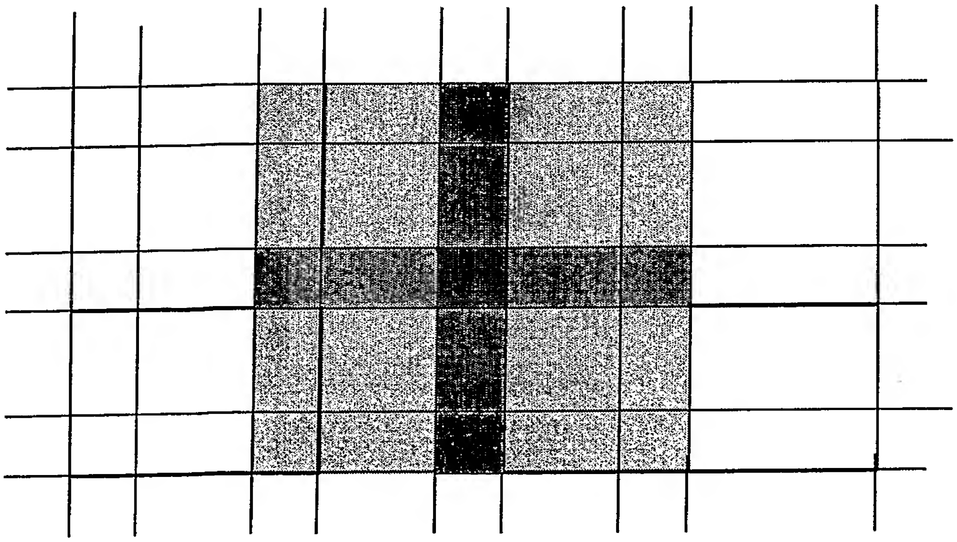


Fig. 13

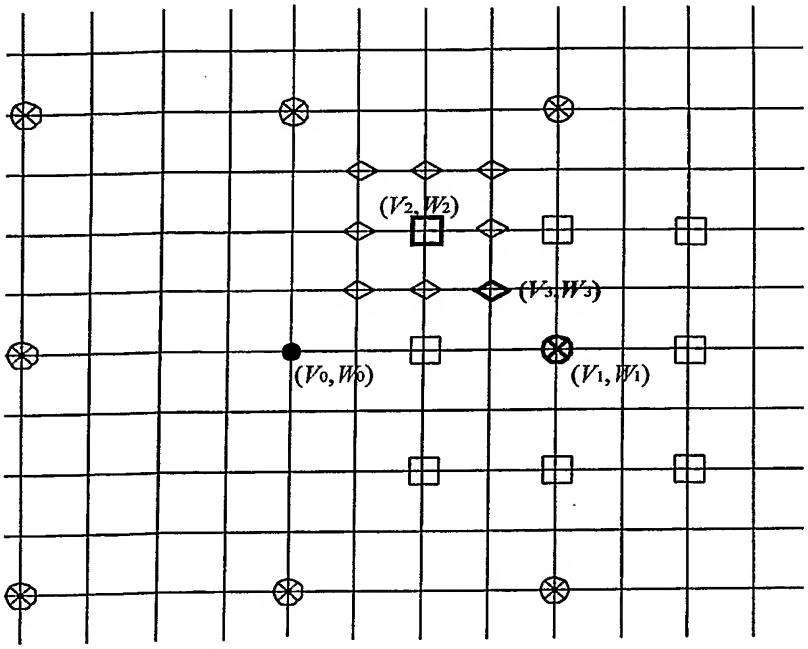


Fig. 14

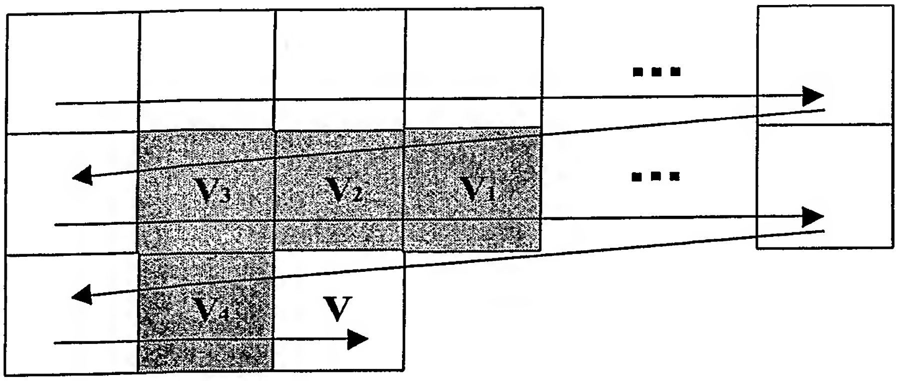


Fig. 15

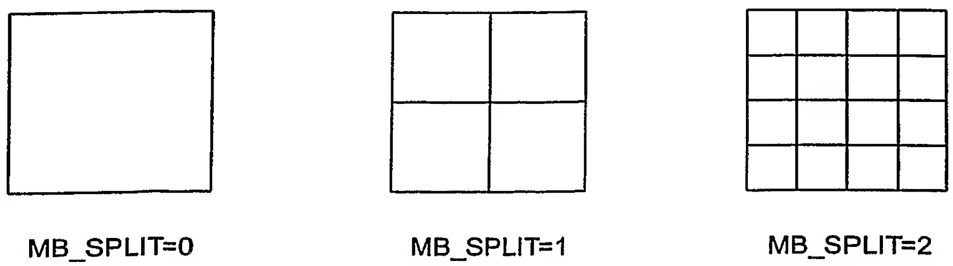


Fig. 16

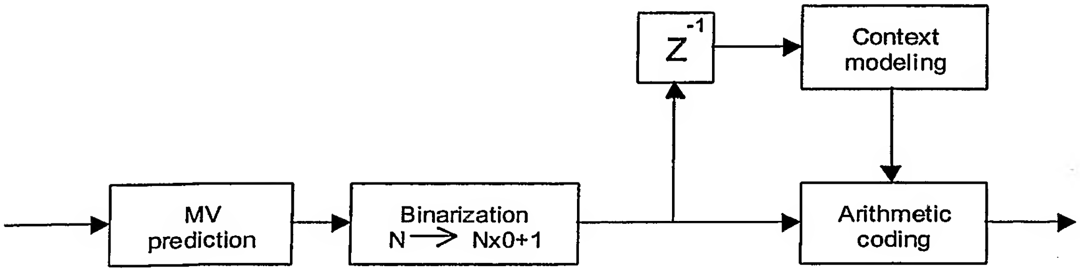


Fig. 17

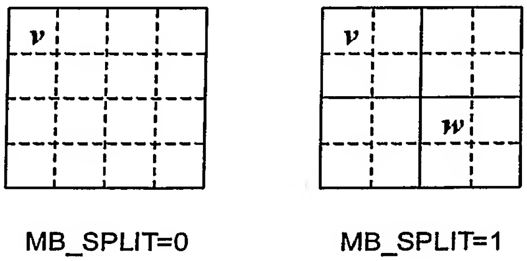


Fig. 18

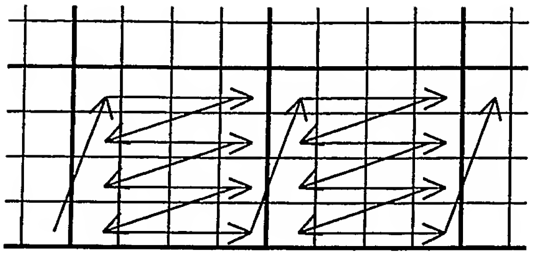


Fig. 19

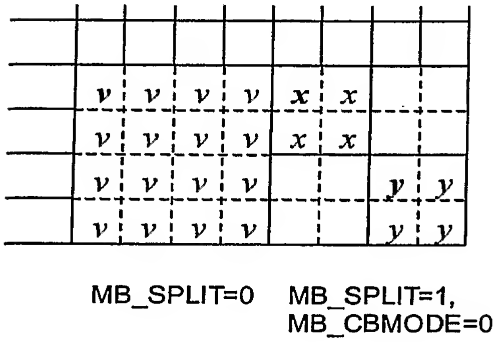


Fig. 20

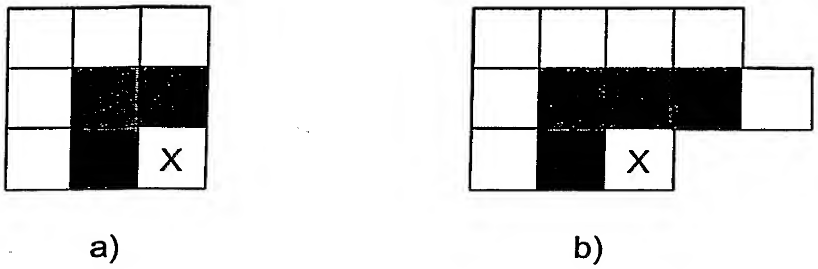
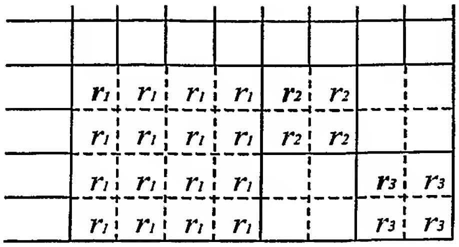


Fig. 21

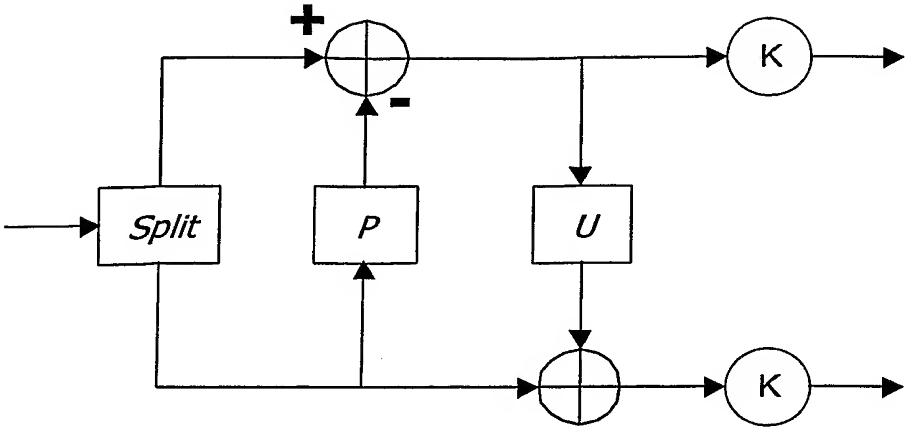


MB_SPLIT=0 MB_SPLIT=1,
MB_CBMODE=0

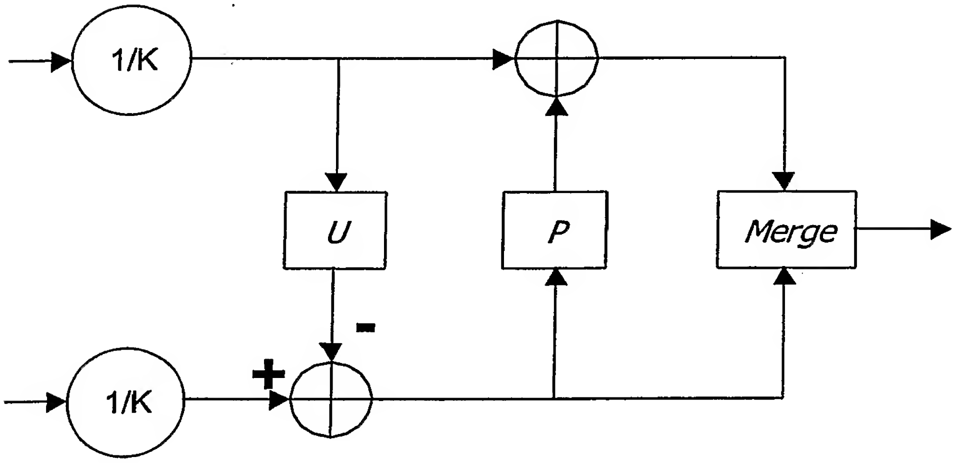
Fig. 22

V(0)	=								1
V(1)	=						0	1	0
V(2)	=						0	1	1
V(3)	=				0	0	1	0	0
V(4)	=				0	0	1	0	1
V(5)	=				0	0	1	1	0
V(6)	=				0	0	1	1	1
V(7)	=		0	0	0	1	0	0	0
V(8)	=		0	0	0	1	0	0	1
V(9)	=		0	0	0	1	0	1	0
V(10)	=		0	0	0	1	0	1	1
V(11)	=	...							

Fig. 23



a) the lifting scheme analysis process



a) the lifting scheme synthesis process

Fig. 24

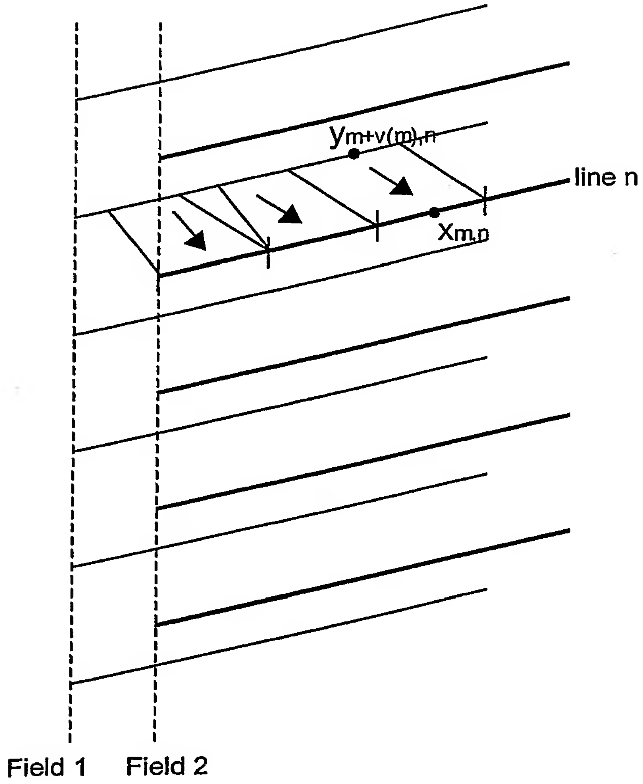


Fig. 25

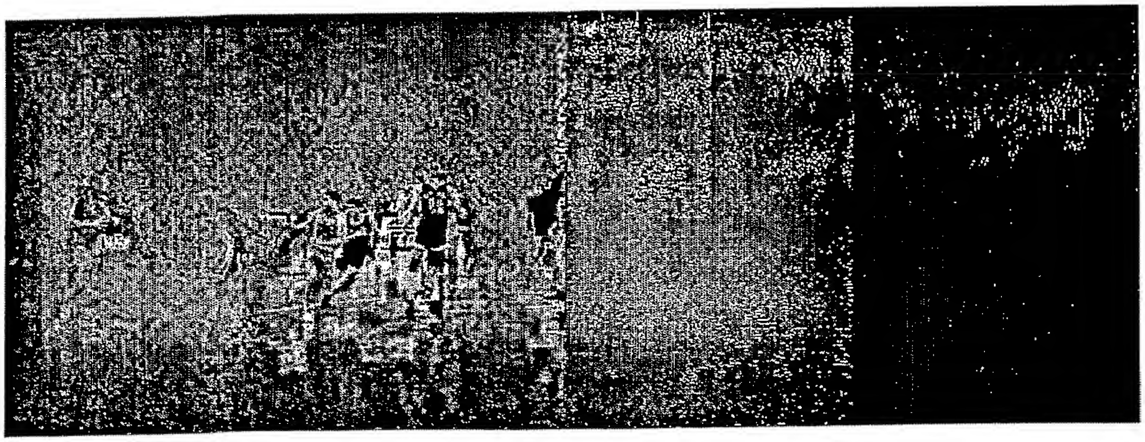


Fig. 26

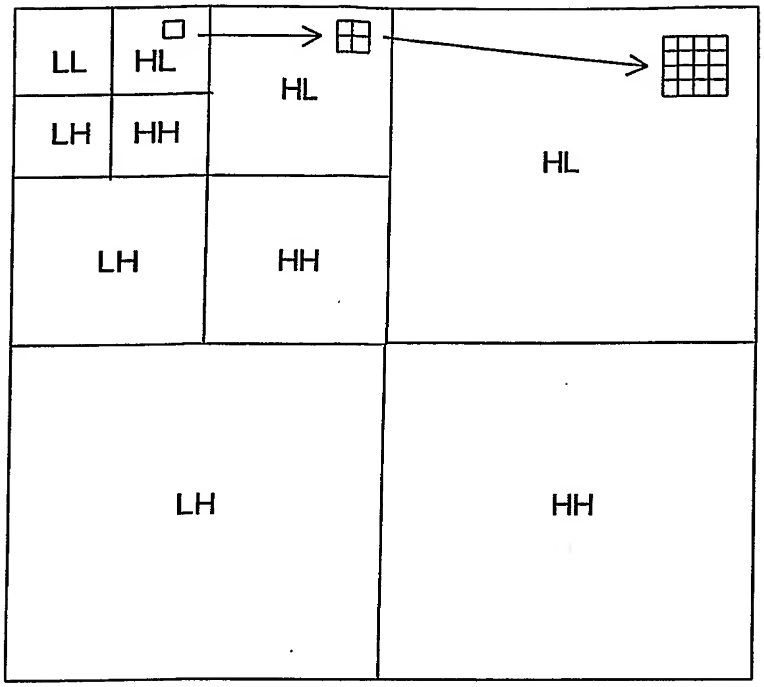


Fig. 27

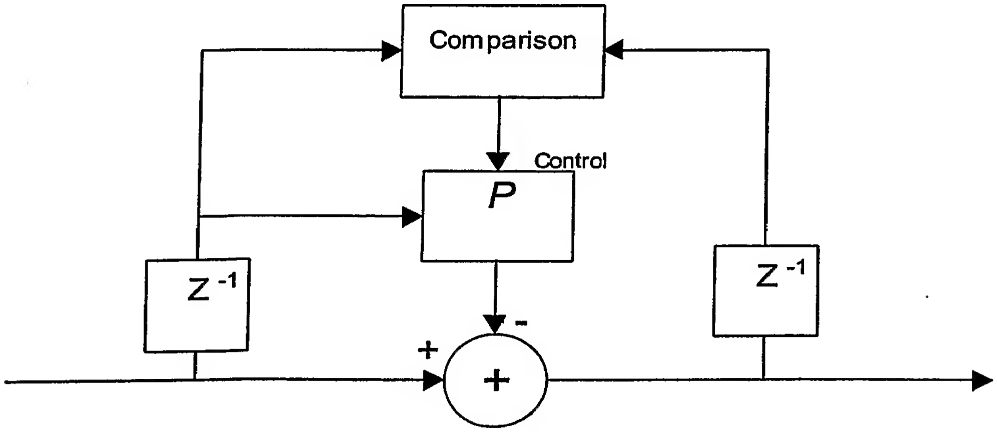


Fig. 28

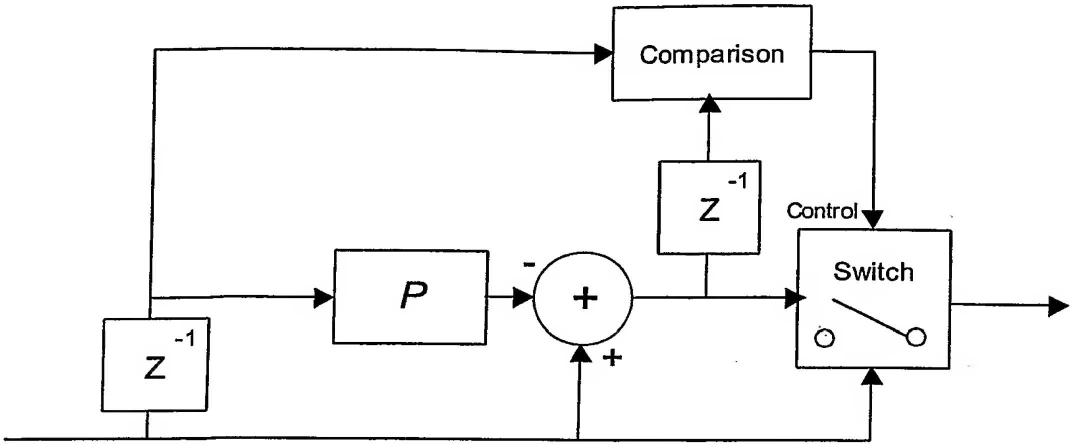


Fig. 29

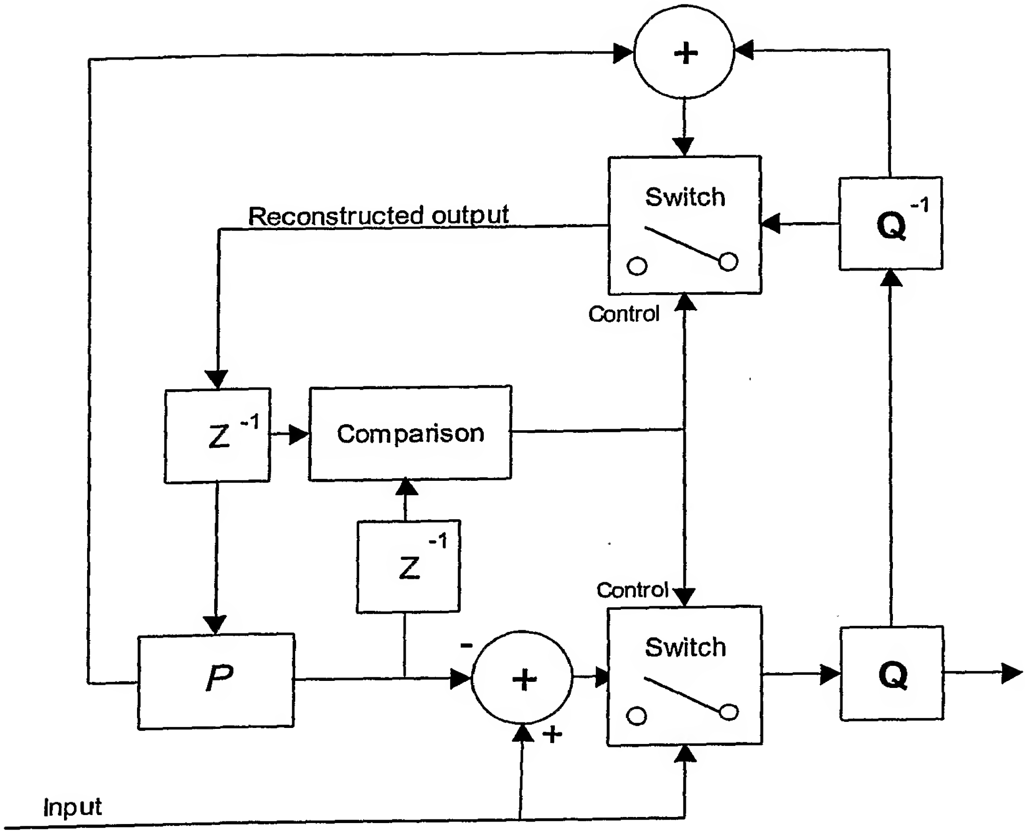

+
Switch
Reconstructed output
Q^{-1}
Control
Z^{-1}
Comparison
Z^{-1}
Control
Switch
P
-
+
+
Q
Input

Fig. 30

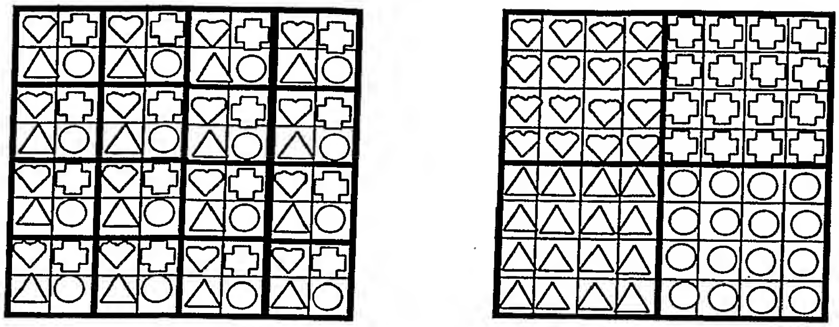


Fig. 31

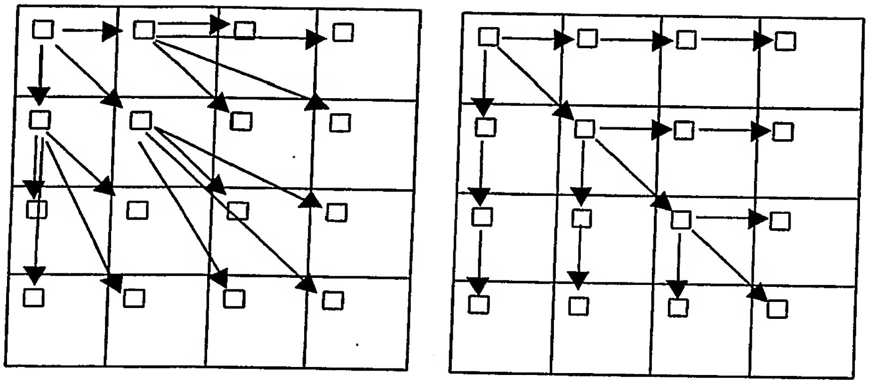


Fig. 32

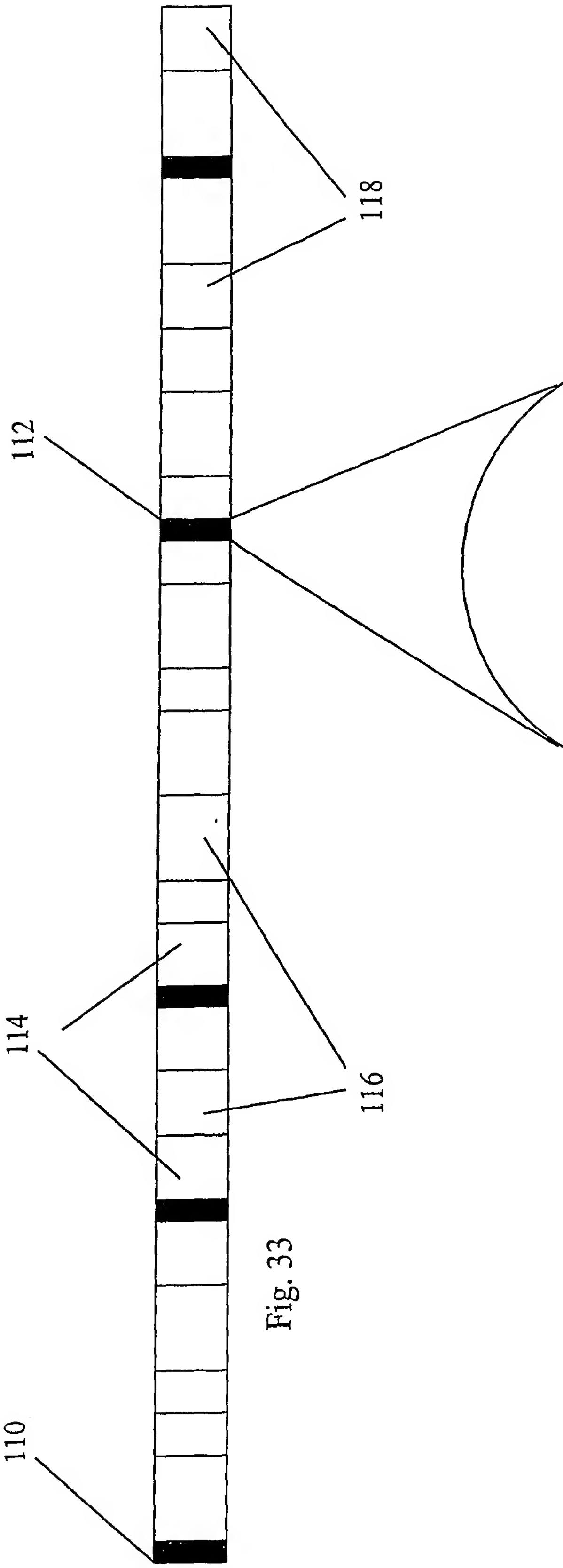


Fig. 33

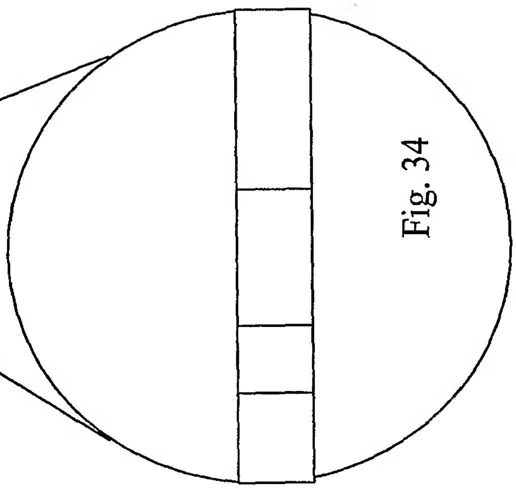


Fig. 34